Multi-Objective Constraint Satisfaction for Mobile Robot Area Defense

THESIS

Kenneth W. Mayo, Second Lieutenant, USAF

AFIT/GCE/ENG/10-03

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/GCE/ENG/10-03

Multi-Objective Constraint Satisfaction for Mobile Robot Area Defense

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Engineering

Kenneth W. Mayo, B.S.C.E.

Second Lieutenant, USAF

March 2010

AFIT/GCE/ENG/10-03

Multi-Objective Constraint Satisfaction

for Mobile Robot Area Defense

Kenneth W. Mayo, B.S.C.E.
Second Lieutenant, USAF

Approved:

//signed//	12 Mar 2010
Dr. Gilbert L. Peterson Chairman	Date
//signed//	12 Mar 2010
Lt Col Brett J. Borghetti, PhD Member	Date
//signed//	12 Mar 2010
Maj Michael J. Mendenhall, PhD Member	Date

AFIT/GCE/ENG/10-03

Abstract

In developing multi-robot cooperative systems, there are often competing objectives that need to be met. For example in automating area defense systems, multiple robots must work together to explore the entire area, and maintain consistent communications to alert the other agents and ensure trust in the system. This research presents an algorithm that tasks robots to meet the two specific goals of exploration and communication maintenace in an uncoordinated environment reducing the need for a user to pre-balance the objectives. This multi-objective problem is defined as a constraint satisfaction problem solved using the Non-dominated Sorting Genetic Algorithm II (NSGA-II). Both goals of exploration and communication maintenance are described as fitness functions in the algorithm that would satisfy their corresponding constraints. The exploration fitness was described in three ways to diversify the way exploration was measured, whereas the communication maintenance fitness was calculated as the number of independent clusters of agents.

Applying the algorithm to the area defense problem, results show exploration and communication without coordination are two diametrically opposed goals, in which one may be favored, but only at the expense of the other. This work also presents suggestions for anyone looking to take further steps in developing a physically grounded solution to this area defense problem.

Acknowledgements

I would like to thank my family and friends for their support and my committee for their guidance.

Kenneth W. Mayo

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Abbreviations

Abbreviation		Page
MAS	multi-agent system	7
CSP	constraint satisfaction problem	9
EA	evolutionary algorithm	13
MOEA	multi-objective evolutionary algorithm	17
NSGA	non-dominated sorting genetic algorithm	17
NSGA-II	non-dominated sorting genetic algorithm II	17
jMetal	Metaheuristic Algorithms in Java	20
JNSGA2	Java non-dominated genetic sorting algorithm II	20
MANET	mobile ad-hoc network	23
MDP	Markov decision process	25
POMDP	partially observable Markov decision process	26
DEC-POMDP	decentralized partially observable Markov decision process	26
I-POMDP	interactive partially observable Markov decision process	27

Multi-Objective Constraint Satisfaction

for Mobile Robot Area Defense

I. Introduction

Dull, dangerous, or extremely precise jobs often utilize robotic systems in place of a human worker [51]. Robots can fill prescriptions at a pharmacy faster and with more precision than a tired or bored human worker. They move radioactive material in power plants and weapons manufacturing facilities without the risk of exposure or poisoning. They manufacture parts of everything in our daily lives, from cars to cell phones to toasters, with a precision unequaled by the most skilled human craftsman.

One application of robotics is to fully automate area and building defense to replace human resources with more expendable systems. These systems need to perform all of the tasks as their human counterparts, including maintaining communication, achieving tasks, and maintaining a current view of the environment. This entire defense system relies on many smaller components to operate, including good communication and trust between the agents in the cooperative system. Good communication is also important so that the users responsible for the automated system can trust the decisions made by the defense system and have an updated status of the system.

Area defense and exploration is the primary goal of the system, but without trust nothing can be accomplished. An agent could become compromised through both hardware (physical damage or destruction) and software (reprogramming via hacking) which could make its contribution to the system unreliable, or even allow the agent to fully subvert the goals of the rest of the system.

Common stochastic problem representations for multi-agent task problems are

based on the decentralized partially observable Markov decision process (DEC-POMDP) [43]. This specification provides the description of the agents, the environment, and the agent interactions with each other and the environment. Solutions for each state are policies which describe the expected actions an agent may take at a particular time step or environmental state. DEC-POMDP policies allow opponent modeling in which an agent can take into account the actions of the other agents in order to gain the greatest reward. However, policy solving algorithms only solve for one objective, maximizing expected utility, where the expected utility is a numerical representation of the reward a certain action or series of actions provides [41].

1.1 Problem

One of the many potential applications of distributed robotics is in area defense. A system of autonomous or semi-autonomous robots are deployed in a geographic location and perform different tasks. These tasks can range from simple area exploration and mapping, to maintenance of a communications grid, to defense tasking and engaging enemies on a battlefield. An important element of this defense system is the trust that exists between all agents and their operators in the environment. Humans model their trust of one another using past events and those events' correlation to desired and intended outcomes. There are many ways to do this with machines as well.

For example, the TI-POMDP (trust-based interactive partially observable Markov decision process) trust model [45] allows each agent to use opponent modeling and observations to determine how trusted the other agents in the system are, and whether they should cooperate with or disregard the actions of the other agents. This model depends heavily on communication between the agents for decidability. Strong communication is easily achieved and maintained using a POMDP-based model by keep-

ing the agents in physical proximity to each other.

This, however, is only one part of the area defense system as a whole. The other important element is exploration of the space to allow the robots in the system to control an area.

This TI-POMDP-based trust system on its own does not allow for the distributed exploration or defense that a multi-agent area defense system needs. To maintain trust agents must stay close to each other and maintain communication, but by doing this they may never achieve their other goal of exploration of the space. This creates a system with two opposing objectives and a need for a multi-objective policy solver.

1.2 Methodological Approach

This research formulates a constraint satisfaction problem (CSP) to statically solve for the goals of exploration and communications maintenance in their relationship to one another, and shows how agents will naturally place themselves to maximize their observability of the environment, as well as maintaining a communications link when the agents are unable to coordinate their actions. The ultimate goal is to provide some baseline for further development of a more complex exploration and communications maintenance protocol.

This research shows that the objectives of area exploration and communication maintenance cannot coexist without more structured planning of agent movements, and that given no other goal or prioritization, agents should naturally position themselves at the limits of their observable ranges in order to maintain both goals. Any coordination or task planning between the agents should utilize this information.

1.3 Significance and Limitations

This research, sponsored by the Air Force Office of Scientific Research (AFOSR), looks to develop a multi-agent system in which multiple conflicting goals can be considered and used to make decisions. It discusses the challenges present when working in a large multi-agent and multi-objective environment.

The biggest limitation of this research is its static nature of presenting the constraint satisfaction problem. Agents cannot move on their own, nor can they model other agents, so this information must be used within the context of a much larger and more realistic system. This thesis makes assertions about using evolutionary algorithms with POMDP solvers which should assist in any other researcher looking to pick up where this research leaves off.

1.4 Outline

The following describes the use of the genetic algorithm NSGA-II in solving for static agent positions in an environment which mesh exploration and communication, with an emphasis on the need for full distribution in any decision making regarding area defense. It covers both the original intent of the research and its challenges, as well as the final product, and the errors made by the researcher that led to that final product.

This thesis consists of six chapters including this introductory chapter. Chapter 2 consists of a literature review which covers the background necessary for accurately defining and solving a multi-objective problem using genetic algorithms, as well as explanations of constraint satisfaction problems. Chapter 3 describes the process used to develop a solution for this particular multi-objective problem in both its original intended form and the actual end product. Chapter 4 explains the testing and validation methods and also presents the results and analysis of results. Chapter

5 gives a conclusion of this multi-objective work and outlines future research.

II. Literature Review

Rarely is it acceptable or efficient in the real world to remain focused on a single task. To get anything of real importance completed in a timely manner, goals must be interlaced and balanced in order to achieve an applicable and useful solution. The area defense problem in particular has to both explore the environment, as well as maintain communication among the system members.

This chapter describes both the goals of multi-objective optimization, as well as aspects of properly modeling the goals for this problem. This includes the description of a wireless communication environment, the definition of trust and its importance in facilitating reliable multi-agent operations, and multi-objective optimization with popular evolutionary algorithms.

2.1 Agents and Multi-Agent Systems

An agent is anything that can sense its environment and act on that environment. Four common agent types are [41]:

- Simple reflex agents - These agents react based solely on their current perception of the environment. These agents are often the most primitive, but can be the fastest to respond to a change in the environment if a rule set is well-enough defined and handled. They ignore everything they cannot observe and do not maintain a history or model of the environment.

- Model-based reflex agents - These agents attempt to model their environment, and possibly the other agents in the system. They use the model to predict what is hidden from their observation and use that to make decisions about the next best course of action.

- Goal-based agents - These agents attempt to use descriptions of a desirable outcome or outcomes to decide what the best course of action is. They combine their knowledge of the environment as well as their desires of how they want the environment to be some time in the future to make their decisions.

- Utility-based agents - These agents use a "happiness" descriptor to decide how good or bad a situation is. Utility allows for a non-binary way of deciding if one state is more preferable than another by mapping the state onto a real number or series of numbers that can then be compared for action determination.

All of these agent types can be extended into learning agents which can make improvements on their behavior rules, goals, or utilities to better improve performance on a specific metric.

A multi-agent system (MAS) is a system in which two or more agents interact with one another and their environment to achieve cooperation, coordination, and negotiation in their actions towards a goal or objective [52]. These agents may or may not be working towards the same goal, and may in fact be working to subvert the other's objectives.

Some researchers have worked to develop standardizable taxonomies for describing the MAS [7, 11, 50]. These have yet to become standards, but they allow for formal definition of most multi-agent environments.

Dudek, et al., [11] describe their MAS by first identifying the problem statement and the number of agents desired to complete the task. Problems range from those which are traditionally multi-agent to those which may benefit from the use of multiple agents (as opposed to one large or much more capable agent). They then define the size of the agent collective, the values of communication range, topology, cost, and reconfigurability (how dynamic the communication topology can be) relative to how important they are to each other and solving the problem. Agents are defined by their

processing power and their uniformity of capability and function [11]. This allows for a formal definition of agent collectives with respect to both the developer's original intent and the formal taxonomy.

Cao, et al., [7] describe an MAS based on the task to be undertaken and its inherent needs. They then define the mechanisms of cooperation, to include the group structure/hierarchy, potential resource conflicts, the origins of cooperation which will facilitate cooperation among the agents, agent learning, and the geometric problems that arise from the operation of agents in two and three dimensional environments. The system is then described by its centralization: how centralized or decentralized the agents and their decision mechanisms are, as well as the homogeneity or lack thereof in the collection of agents (called differentiation).

Weiss [50] describes a MAS by first defining a communications protocol to facilitate in agent coordination. This description of relies on the assigned abilities of each agent in the system to participate in the communication. Agents can be basic in that they only receive information, passive in that they only send and receive information, active in that they send and receive information and also ask for information, and peers in that they send and receive information, as well as receiving and responding to queries for information. Weiss' MAS taxonomy is unique in that the communications protocol and syntax structure which allow for coordination and belief maintenance is of the utmost importance in describing the system, and it is only after this is strongly defined that the specific problem is considered.

There exists commonality between the above formalisms, but as can be seen, there is no universally accepted standard for describing or solving a multi-agent problem.

All multi-agent systems allow for the completion of tasks that may not be achievable by a single agent system. A single agent may not be physically capable of completing a task individually or it may not have the information necessary to complete

the task, therefore needing the information contained within other agents in the system. Common multi-agent research includes communication [23], fault-tolerance [29], and organization [42]. Multi-agent systems are commonly found in graphical applications such as video games and often applied to area defense like problems in which a single physical entity is not enough to achieve the desired goal.

2.2 Constraint Satisfaction Problems

A constraint satisfaction problem (CSP) is a problem which is defined by a set of variables which must meet certain requirements in order to be considered a proper solution [41].

A CSP is formally defined as a set of variables:

$$\{V_1, V_2, ..., V_n\},$$

a set of constraints:

$$\{C_1, C_2, ..., C_m\},$$

and a set of domains of values for which each variable can take on:

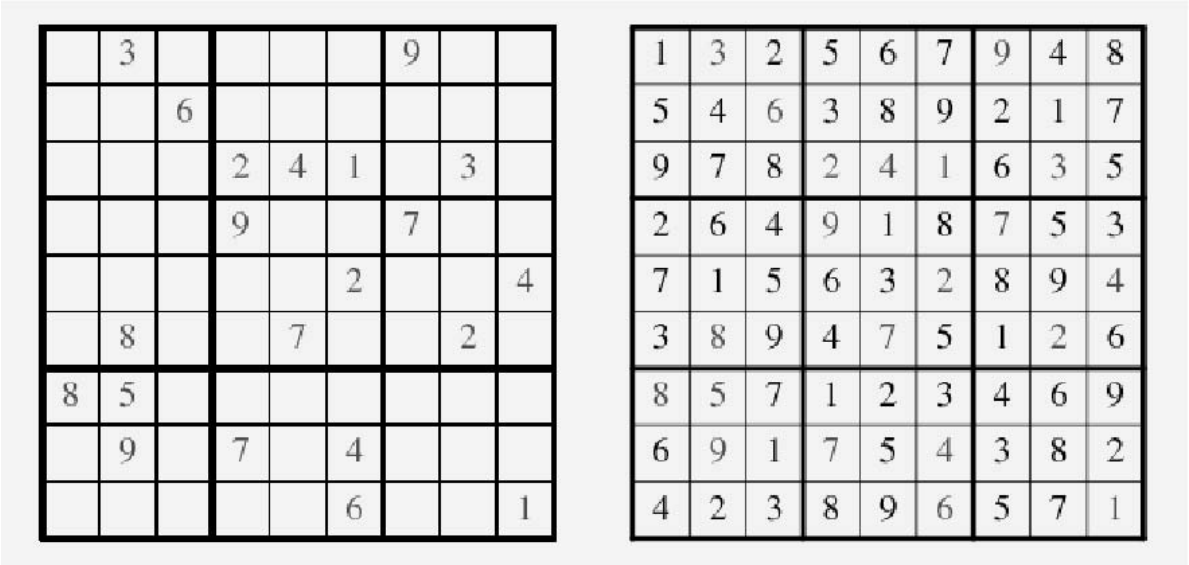
$$\{D_{11}, D_{12}, ..., D_{1x}\}, \{D_{21}, D_{22}, ..., D_{2y}\}, ..., \{D_{n1}, D_{n2}, ..., D_{nz}\}.$$

The variable V_a is of a non-null value in the domain D_{ai}. Each constraint consists of a subset of variables and constraints which mathematically define an allowable combination for that subset. A state of the problem is defined by an assignment of values to some or all of the variables

$$\{V_i = v_i, V_j = v_j, ...\}$$

and is considered legal if it does not violate any of the constraints. A solution to a CSP is a complete assignment set for the variables that meets all constraints [41]. There can often be many legal solutions to a single CSP.

A common and popular puzzle is the Japanese game Sudoku [22], shown in Figure 1, in which a player is given a partially empty grid of numbers and required to assign a numerical value between 0 and 9 to all of the empty spaces such that the game's constraints are satisfied. A standard game of Sudoku consists of a 9x9 board with 9 3x3 subgrids. Each 3x3 subgrid must contain the numbers 0 through 9 with no repetition. Along with filling the subgrids, each horizontal row and vertical column in the full 9x9 board must contain the numbers 0 through 9 only once.



	3					9		
		6						
			2	4	1		3	
			9			7		
					2			4
	8			7			2	
8	5							
	9		7		4			
					6			1

1	3	2	5	6	7	9	4	8
5	4	6	3	8	9	2	1	7
9	7	8	2	4	1	6	3	5
2	6	4	9	1	8	7	5	3
7	1	5	6	3	2	8	9	4
3	8	9	4	7	5	1	2	6
8	5	7	1	2	3	4	6	9
6	9	1	7	5	4	3	8	2
4	2	3	8	9	6	5	7	1

Figure 1. An incomplete Sudoku game (left) and its completed counterpart (right). Aside from the standard game rules, the empty board provides additional starting constraints in red which the remaining variable assignments must meet (shown in black) [26].

As a CSP, a traditional Sudoku game would be defined as the variables

$$\{E_1, E_2, ..., E_n\}$$

representing each of the empty spaces on a board, and the constraint set

$$\{S_1, S_2, ..., S_9, R_1, R_2, ..., R_9, C_1, C_2, ..., C_9\}$$

representing each of the 3x3 squares, the 1x9 rows, and the 9x1 columns on the board. This Sudoku game could then be solved using one of the many CSP solving methods, described in the next section.

A more traditional CSP is the "N-queens problem", shown in Figure 2, in which N chess queens are placed on an $n \times n$ chessboard in a way that no two queens can attack each other, that is, each queen occupies its own individual row, column, and diagonal. Defined as a CSP [47], the variable set is

$$\{Q_1, Q_2, ..., Q_N\}$$

with the problem constraints

$$\{Q_i \neq Q_j, |Q_i - Q_j| \neq |i - j|\}.$$

2.3 Methods for Solving Constraint Satisfaction Problems

A CSP is best solved using a local search method [41] in which only improvements that exist a certain distance away (defined by the problem) from the current location are considered. Common methods involve deterministic solvers for the search, either with or without backtracking, forward checking, and minimum conflict search [41].

Search with backtracking involves assigning values to the variables one at a time, keeping track of their legality within the problem constraints. When an illegal assignment is made, the search backs up to the last known legal configuration and discards

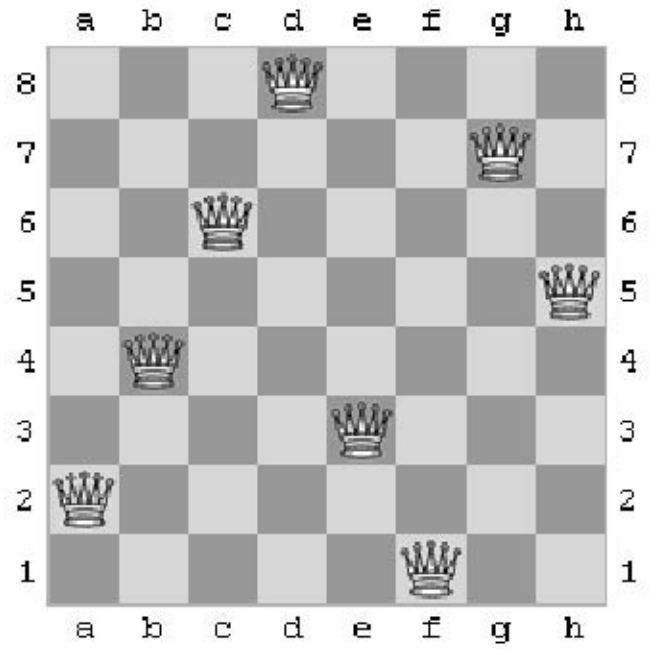


Figure 2. One of the many unique solutions for the N-queens problem where N = 8 [39].

all possible solutions that could have been generated from that illegal assignment. This type of search is a form of admissible brute force, which can be improved by using a heuristic to keep track of the least constrained value, and using it as an assigned ordering [41].

Search with forward checking keeps track of the remaining legal values for each of the variables, and terminates when a variable has no more legal values. This is a form of inadmissable local constraint enforcement and does not provide any early detection for illegal assignments. When an illegal assignment is detected, it becomes necessary to revert to a previous state in which there remained legal assignments to be made [41].

Minimum conflict search (min-conflicts) assigns random domain-consistent values to each variable of a CSP. It then iteratively deconflicts the values to achieve legality with the constraints by assigning each variable to the value that has the fewest con-

flicts with the constraints. The search ends when a maximum number of iterations has been met, or a legal solution has been found [41].

Because of the local search nature of the problem, many population-based stochastic search algorithms, such as an evolutionary algorithm (EA), can be used to solve CSPs [13]. In using an EA to solve a CSP, the CSP must be described in a fashion consistent with the requirements of an EA. Typically an individual in the evolutionary algorithm's population is the complete set of variable assignments. The EA's fitness functions are used in lieu of a set of constraints to determine the legality of a set of variable assignments and the variable assignments themselves will be determined with some random probability. This method for solving is inadmissable, but is often faster than a deterministic search [13, 40].

2.4 Evolutionary Algorithms

The evolutionary algorithm (EA) concept works to include stochastic elements from biological evolution in the development of an optimal solution. The EA optimizes a solution by selecting an outcome from a population of possible outcomes which are manipulated using traditional elements from biological evolution [8]. EAs function around consecutive generations of solution sets and typically include the following basic steps [14]:

- Population Generation - a set of possible solutions (the population) is needed at the start of the algorithm. This population of solutions is created from a seed, a predetermined set of values or information which is changed (mutated) to create the population. This is also called "crossover".

- Fitness Evaluation - of the current population, each solution candidate is weighed and scored objectively to determine its fitness as a final solution.

- Selection - of the current population, the solutions with the strongest candidacy for becoming a final solution are combined into a single solution with each "parent" contributing at random to the "offspring", and the offspring is used as the seed for the next population generation.

- Crossover - the population members chosen from the selection stage are combined in a user-defined way to generate parts of a new population.

- Mutation - the seed from the previous generation is altered, with large or small changes performed at the programmer's discretion to create a population of potential solutions for the next generation.

- Termination - this is the condition under which the algorithm will discontinue can either be an acceptable solution or a specified number of generations.

Mutation is also shown in Figure 3 with a binary string representing the genome and a string representing the desired solution. The elements in red represent the randomly selected variables to be mutated. There are three population members that are equally close to the desired solution, so the first one found is used as the seed for the next generation.

Each stage of this process, with the exception of the fitness evaluation and the process termination, occurs with random probability and there is a chance for poor solutions to propagate through the entire system and produce no useful result, therefore, it is the responsibility of the fitness evaluation and selections stages of the EA to push for quality solutions. Fitness for a particular solution may either be compared against a static known or wanted solution, or as is usually the case, compared against an objective function [14]. A stochastic measure may be applied to the selection process such that better solutions have a greater chance of being selected, but weaker solutions also have a chance of moving on to the next generation.

Seed	0	1	1	0	0	1	0	1
Population	1	1	1	0	0	1	1	1
	0	1	0	0	0	0	0	1
	0	0	1	0	1	1	0	1
	1	1	0	0	0	1	0	1
	0	1	1	1	0	1	0	0
	0	1	1	0	0	0	0	0
	0	0	0	0	0	1	0	1
	0	1	1	0	1	1	1	1
Desired	0	0	0	0	0	0	0	0
New Seed	0	1	0	0	0	0	0	1

Figure 3. A binary seed mutation for a generic EA.

Thomas Bäck defines each stage of the generic EA mathematically with the generic EA defined as

$$EA = (I,\Phi,\Omega_{EA},\Psi,s,\iota,\mu,\lambda)$$

where $I = A_x \times A_s$ is the population itself, and A_x, A_s are sets of variables and their assignments, respectively. $\Phi : I \to IR$ is a fitness function which assigns real values to population members.

$$\Omega_{EA} = \{\omega_{\theta 1}, ..., \omega_{\theta z} | \omega_{\theta i} : I^\lambda \to I^\lambda\} \cup \{\omega_{\theta 0} : I^\mu \to I^\lambda\}$$

is the set of probabilistic genetic operators $\omega_{\theta i}$, each of which is controlled by specific parameters in the sets $\Theta_i \subset IR$. The operator

$$s_{\Theta s} : (I^\lambda \cup I^{\mu+\lambda}) \to I^\lambda$$

defines how many population members are selected. This can change from λ to $\lambda + \mu$ to μ, where $\mu, \lambda \in IN$ and $\mu = \lambda$ is allowed. Θ_s may be an additional selection operator, μ is the number of parents, λ denotes the number of offspring in each

generation, $\iota : I^{\mu} \rightarrow \{true, false\}$ is the termination criterion for the algorithm, and $\Psi : I^{\mu} \rightarrow I^{\mu}$ is the complete process of transforming a population P into the next by applying the genetic operators $\Psi = s \circ \omega_{\Theta i1} \circ ... \circ \omega_{\Theta ij} \circ \omega_{\Theta 0}$ and $\Psi(P) = s_{\Theta s}(Q \cup \omega_{\Theta i1}(...(\omega_{\Theta ij}(\omega_{\Theta 0}(P)))...))$ where $\{i_1, ...i_j\} \subseteq \{1, ...z\}$ and $Q \in \{\emptyset, P\}$ [4].

2.5 Multi-Objective Solvers

Multi-objective solving is the process of optimizing multiple distinct and potentially conflicting goals [8]. For example, a group of robots has to defend an area and maintain network communication. In order to properly defend the area, the robots must spread out far enough to cover the space, but in doing so may move out of communication range with the other robots. In order to regain communication, the robots must move closer together and possibly defend less area. Multi-objective optimization is concerned with maximizing goals in the way that least impacts the other desired goals.

There are multiple methods for representing a multi-objective problem for solving such as

- Aggregation - This is the simplest method for solving multiple objective problems. It involves combining each of the goals into single or multiple goal functions, with each sub-goal weighted according to its desired impact on the final solution. These weights can be static or dynamic between optimization runs [24, 36].

- Population-based - The population-based approach includes elements from biology such as mutation, combination, and selection, in which there exists a probability that two solutions from a given population of solutions will be combined, a solution will be altered randomly, or a lower ranked solution will be

chosen over a higher one. This can allow for multiple solutions to be combined in a way that improves the overall solution [16].

Multi-objective solving research is currently in the realm of the population-based approach, with single and multiple aggregation popularly seen as inferior and used as a crude performance comparison.

2.5.1 Non-Denominated Genetic Sorting Algorithm II (NSGA-II).

A multi-objective evolutionary algorithm (MOEA) is an evolutionary algorithm in which the fitness function (Φ) and evaluation is geared towards the requirements of two or more (possibly independent) goals, making $\Phi : I \to \mathbb{R}^k$ where $k \geq 2$ [8].

One of the first MOEAs used the number of dominating elements to rank population fitness [8]. This was later expanded into the non-dominated sorting genetic algorithm (NSGA) which used multiple layers of classification to describe each population element. This allowed for a better search of the Pareto front regions and allowed the population to converge near them [10]. NSGA, while being very good at convergence, suffered from three main weaknesses:

1. $O(M\lambda^3)$ complexity, where M is the number of objectives and λ is the population size.

2. Non-elitism in which weak solutions are given the same chance to survive to the next generation as their strong counterparts - no survival of the fittest.

3. The need to specify a sharing parameter to guarantee the diversity of solutions in the next generation.

These weaknesses were all addressed in NSGA-II. Pseudocode for NGSA-II is shown in Algorithm 1 [8]. NSGA-II has a complexity of $O(M\lambda^2)$ due to its faster non-dominated sorting algorithm, uses an elitist approach in which the more fit solutions

are more likely to be used as seeds for the next generation, and defines a crowding distance in its selection operation to keep the population diverse [10]. NSGA-II is currently the standard for comparison in most MOEAs [8].

Algorithm 1 NSGA-II
from *Evolutionary Algorithms for Solving Multi-Objective Problems*

1: **procedure** NSGA-II $(\lambda', g, f_k(x_k)) \triangleright \lambda'$ members evolved g generations to solve $f_k(x)$
2: Initialize Population
3: Generate random population - size
4: Evaluate Objective Values
5: Assign Rank (level) Based on Pareto Dominance - *sort*
6: Generate Child Population
7: Binary Tournament Selection
8: Recombination and Mutation
9: **for** $i = 1$ to g **do**
10: **for** each Parent and Child in Population **do**
11: Assign Rank (level) based on Pareto - *sort*
12: Generate sets of nondominated vectors along PF_{known}
13: Loop (inside) by adding solutions to next generation starting from the *first* front until N' individuals found determine crowding distance between points on each front
14: **end for**
15: Select points (elitist) on the lower front (with lower rank) and are outside a crowding distance
16: Create next generation
17: Binary Tournament Selection
18: Recombination and Mutation
19: **end for**
20: **end procedure**

2.5.2 Pareto Optimality.

Pareto optimality plays a big part in multi-objective solvers, especially in the ranking of EA-based solutions as it is usually how solutions are ranked against each other. Pareto optimality is an economics and game theory concept in which distinct goals can be optimized in a multiple objective environment. A Pareto optimal solution

is one in which that particular solution can no longer be improved without impacting the effects of any of the other goals' solution [8].

It is said that one element strongly Pareto dominates another element if it is better in all of its desired characteristics. An element is non-dominating of another element if it is better in one aspect but not another. The optimal set of points are all those which are non-dominated by each other; they form the Pareto frontier which is the series of points in which the definition of "better" or "stronger" or "more correct" becomes ambiguous.

Formally, Pareto terminology is defined as [8]

- Pareto Optimality: A solution $\mathbf{x} \in \Omega$ is said to be Pareto Optimal with respect to Ω if and only if there is no $\mathbf{x}' \in \Omega$ for which $\mathbf{v} = F(\mathbf{x}') = (f_1(\mathbf{x}'), ..., f_k(\mathbf{x}'))$ dominates $\mathbf{u} = F(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_k(\mathbf{x}))$.

- Pareto Dominance: A vector $\mathbf{u} = (u_i, ..., u_k)$ is said to dominate another vector $\mathbf{v} = (v_i, ..., v_k)$ (denoted by $\mathbf{u} \preceq \mathbf{v}$ if and only if $\mathbf{u}$ us partially less than $\mathbf{v}$, i.e. $\forall i \in \{1, ..., k\}, u_i \leq v_i \wedge \exists i \in \{1, ..., k\} : u_i < v_i$.

- Pareto Optimal Set: For a given multiobjective problem, $F(\mathbf{x})$, the Pareto Optimal Set, $\mathcal{P}^*$, is defined as: $\mathcal{P}^* := \{\mathbf{x} \in \Omega \mid \neg\exists\, \mathbf{x}' \in \Omega\, F(\mathbf{x}') \preceq F(\mathbf{x})\}$.

- Pareto Front: for a given multiobjective problem, $F(\mathbf{x})$, and Pareto Optimal Set, $\mathcal{P}^*$, the Pareto Front $\mathcal{PF}^*$, is defined as: $\mathcal{PF}^* := \{\mathbf{u} = F(\mathbf{x}) \mid \mathbf{x} \in \mathcal{P}^*\ \}$.

As shown in Figure 4, the optimal solutions (in red) lie along a boundary (the line) in which no one point dominates another, but all the blue points are dominated by one or more point on the frontier. Points A and B do not dominate each other, but they both dominate point C. Points A and B are considered more optimal than point C for the solution of competing goals 1 and 2.

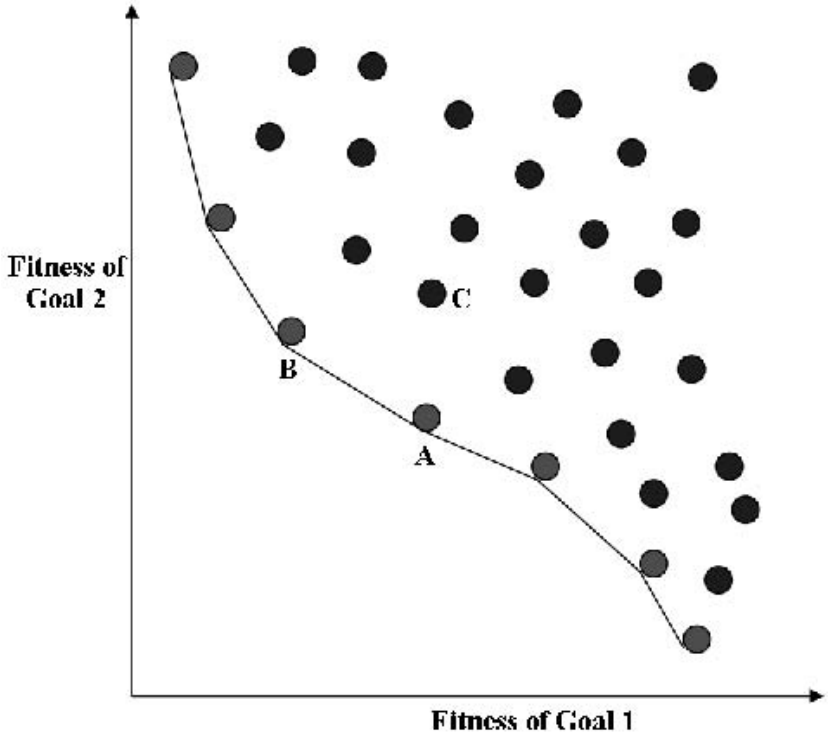


Figure 4. An example of the Pareto frontier for a minimization problem.

The Pareto front can be used to prune away inefficient or unwanted goals to trim the solution space in a way that makes the decision of the proper course of action easier for the agent.

2.5.3 jMetal and JNSGA2.

There currently exist two development libraries for implementing the NSGA-II algorithm in the Java programming language, jMetal [12] and JNSGA2 [33], both of which are open-source under the GNU General Public Licence Agreement.

JNSGA2 is a highly abstract implementation of the NSGA-II algorithm that works to minimize objectives. The developer must create the genetic representation for their individual problem as well as implement the interface for each objective. Mutation and crossover probabilities, desired population size, number of desired generations,

fitness functions, and all other necessary NSGA-II parameters must be implemented by the developer.

jMetal is a development framework geared towards the study of metaheuristics for multi-objective problems. It provides a rich and robust set of object classes which implement 13 different evolutionary algorithms (including NSGA-II). It provides problem families as well as classical and constrained problem examples in an effort to make jMetal as easy-to-use as possible. It allows for explicit variable representation in both real, binary, integer, and binary-coded real numbers.

2.6 Multi-Robot Tasking

In a multi-agent environment, trust implementations (described in a following section) can prevent subversion by enemies, as well as allow for proper opponent modeling. This becomes especially important in the modern environment where communication can be tampered with, and agents themselves can be compromised by non-obvious means.

Much research has been put forward in the realm of group-robot tasking in a real-world environment [20, 37]. As with any multiple component systems, there are multiple categories for multi-robot tasking in an environment:

1. No distribution - the robots themselves receive all their tasking and task de-conflicting from a lead robot or central base station.

2. Partially distributed - the robots are responsible for some or all of the task de-confliction, but must communicate that amongst each other.

3. Completely distributed - the robots can coordinate without direct communication with one another, and sense the environment, the other robots, and act accordingly.

Most of the current research is in the first two categories, with the existence of a base station to communicate mission critical information to the robots in the field or keep their actions properly coordinated [17, 30, 46]. This is less desirable for this particular problem domain, as trusting the tasker or maintaining communications with the tasker is part of the larger challenge to be taken on.

Some work, however, has been done in more distributed multi-robot tasking, including some UAV research in which the UAVs coordinate amongst themselves with limited communication to properly identify a target and gather information [35]. This has also been done on the ground with area defense robots. The roaming robots can decide how to handle a security threat, while maintaining perimeter control, but they also require communication among each other to respond to a threat and a base station to designate their specific patrolling area [20]. Area patrol when communication is very limited is as close to a completely distributed solution as current technology will allow. It has been shown that even when communication is only allowed in specific locations at specific times, robots can still coordinate themselves to gather the most information about their assigned environment [34]. This, however, assumes perfect trust between the agents at all times, and does not consider a situation in which systems have been compromised and agents are intentionally distributing false information to the other agents in the system.

Some tasking analysis has taken place, and at least one formal taxonomy has been developed [17]. This taxonomy assumes that each agent is capable of estimating its own fitness for every task it can perform and begins with defining the utility metrics (U) for task assignment. If an agent A can achieve a task T with a quality $Q_AT = 20$ at cost $C_AT = 10$, and an agent B can achieve the same task with quality $Q_BT = 15$ at cost $C_BT = 5$, then there should be no preference when searching for efficient

agent assignments because

$$U_T = Q_T - C_T$$

and

$$U_A T = 20 - 10 = 10 = 15 - 5 = U_B T \text{ [17]}.$$

This taxonomy then moves to combinatorial optimization and the need to select an algorithm to properly make assignments. It covers subset maximization, in which the most tasks are performed at once, as well as the greedy algorithm for task assignment, in which the least or most expensive tasks are assigned first. Formal measurement metrics are defined for this taxonomy [17], and they are

- single-task agents (ST) versus multi-task agents (MT) which define the differences between agents that can only perform one task at a time and agents that can perform multiple tasks at a time.
- single-agent tasks (SR) versus multi-agent tasks (MR) which define tasks as requiring only a single agent or multiple agents to achieve.
- instantaneous assignment (IA) versus time-extended assignment (TA) which describes an assignment system in which no future planning is considered or one in which future assignments are taken into account.

2.7 Wireless Communication

Strongly coordinated multi-agent activity often depends on reliable communication between the agents in the environment, and it can often be the case that manipulation and observation of the environment is not sufficient and explicit communication methods are needed.

A mobile ad-hoc network (MANET) is a wireless communication network in which

each node of the network acts as both a node and a router; that is, each node in the network can both be the source/destination of information, as well as a forwarding path for other information traversing the network. This allows for flexibility that a fixed-router network does not have in that nodes can enter and leave the network at will and the network mesh can still be maintained.

There exist some important axioms of MANET research that must be maintained throughout development [28]:

- *The world may not be flat.* It is very important when working in a 3-space environment to remember and model proper distance as well as any objects and their radio-blocking properties that may be between nodes.

- *A radio's transmission area is not necessarily circular.* Objects in the environment can block radio signals, as well as other electromagnetic or natural interference.

- *All radios do not have equal range.* A low-power handheld radio will not have the same communications properties as a large television or radio tower.

- *Symmetry in communication may not always exist.* It is not necessarily true that a node which can receive information also has the ability to return information to that sender.

- *Perfect communication is not guaranteed.* It must be remembered that if two nodes are fixed, they may have some stochastic communication ability in that messages come through perfectly, garbled, or not at all with some changing probability.

- *Signal strength is not always a function of distance.* The objects in between the nodes combine with any electromagnetic disturbances in the environment to alter the signal strength and transmission distance of a particular node.

These fundamentals exist in all wireless communication research, but can and are often abstracted away. The particular researcher can also state that their particular hardware or setup corrects for or eliminates these problems. Software packages such as JiST/SWANS [2], NS2 [1], and GloMoSim [53] are often used to precisely model these physical network constraints in simulation, but were not used for this research due to their extremely precise nature. It is not necessary to describe intricate network details, such as network addressing, handshaking protocols, and packet collision.

2.8 Wireless Sensor Networks

This research is directed towards artificial intelligence and agent-based systems, but it is also akin to the placement of wireless sensor networks in an environment, of which much research has been done.

A wireless sensor network is a set of spatially distributed sensors which are used to monitor physical or environmental conditions [27]. These networks can be used for optical sensing, weather sensing equipment, intrusion detection grids, and even agriculture systems [15]. Most of the research done in the setup of wireless sensor networks has been towards their deployment in an environment where intrusion detection is the primary concern [**?**, 25, 55]. These studies vary in their description of the environment. Some consider weather and environmental conditions as well as the time to detect a threat [**?**], while others focus on the lifetime of the network relative to its performance [25]. Some simulations even allow for sensor networks that have a limite capacity to move within their environment [55].

2.9 Markov Decision Processes

There are countless ways to describe an environment and a decision making problem, but the most intuitive is the Markov decision process (MDP), in which all deci-

sions are conditionally independent of all past states.

The Markov decision process [6] is a method for modeling, decision making, and optimizing single-agent domains in fully observable environments where there exists no uncertainty about state.

An MDP is defined by the tuple (S,A,T,R) where

- S is a finite set of environmental states ($S \in s$).
- A is a finite set of actions ($A \in a$) that can be taken.
- T is the set of transition functions $S \times A \rightarrow S'$ that details the probability of an action a changing a current state s to another state s'.
- R is the expected reward that an agent receives from taking an action a in a state s and reaching a another state s'.

The solution to an MDP is a policy ($\Pi(s)$) which defines the action at every possible state that will achieve the greatest reward.

Some of the limitations of the MDP are eliminated by the partially observable Markov decision process (POMDP) [18]. This extension of the MDP is defined by the tuple (S, A, T,Ω,O, R) where S, A, T, and R are unchanged from the standard MDP and

- $\omega \in \Omega$ is a finite set of observations that an agent can make within its environment.
- $o \in O$ is the set of observation functions $S \times A \rightarrow \Pi(\Omega)$ which defines the probabilities of new observations o given that an agent took an action a to enter a new state s'.

The decentralized partially observable Markov decision process (DEC-POMDP) [43] extends the POMDP to a multi-agent environment.

A DEC-POMDP is defined by the tuple $(I, S, A, T, R, \Omega, O, H)$ where S, A, T, R, Ω, and O are amended from the POMDP to include the existence and models of the other agents in the system and

- I is the finite set of agents in the system.
- H is a value that may or may not exist, which represents the positive time horizon for calculations.

The DEC-POMDP has a limitation that it does not allow agents to interact with the environment, that is, the set of environmental states cannot change. The interactive partially observable Markov decision process (I-POMDP) [18, 44] allows any of the agents in the system to interact with the environment and modify the current state and is defined by the tuple $(IS_i, A, T_i, R_i, \Omega_i, O_i)$ for each agent i in the environment and is more analogous to a series of POMDPs for each agent in the system and

- IS_i is the finite set of interactive states $S \times M_j$ that the agent can be in with S as the set of states, and M_j is the set of models of agent j. This includes the changes made to the environment by the other agents in the system in all possible histories h_j.
- A is the set of joint actions of all agents, $A_i \times A_j$.
- T_i is the transition model $S \times A \times S'$ that defines the probability that an agents action a will change the state from s to s'.
- R is the expected reward function $IS_i \times A \rightarrow R$ in which agent i expects to receive r by taking action a into state s.
- Ω_i is the set of observations that an agent can make.

- O is the probability function $S \times A \times \Omega_i$ which defines the probability of an agent taking action a inside state s and producing observations Ω.

An unbounded I-POMDP is undecidable [43], meaning it is too complex to be admissible, and an optimal POMDP is NP-complete. There are many methods to simplify and estimate the computations of the POMDP, DEC-POMDP, and I-POMDP that will not be covered in this work.

2.10 Solving an MDP

Solving a Markov decision process is a computationally complex task as it suffers from dimensional limitations in which the space of all possible solutions is far too large to perform a traditional search through. This manifests itself in a computer simulation as a lack of memory or processing power to effectively solve the problem.

With this, there currently exist techniques for solving MDPs.

- Divide-and-conquer - This computer science approach of breaking a complex task into smaller more manageable ones has been shown useful in solving MDPs. Methods for representing the decision policies for divide-and-conquer include algebraic decision diagrams [49, 31], task graphing, and finite state machines [32]. Divide-and-conquer is a form of dynamic programming, which can also be used to maximize policy or state value to their convergence in multiple iterations.

- Linear programming - The MDP decision space is represented as a mathematical function to be minimized or maximized accordingly. This method requires the state space to be represented as a series of variables, and will work for both discrete and continuous states [21]. This is often the most practical solution type, but can produce an inadmissible solution.

- Prioritized sweeping - The MDP is solved by working backwards from goal

states, placing priority on state variables which change nontrivially from one state back to its previous state [48].

- Equivalence folding - This is a more traditional tree-based search solution in which states that share the same action at the same time are folded together into one state, reducing the search time from that point forward [19].

- Envelope-based - Solving an MDP using an envelope-based solution is traditionally done where the state space is extremely restrictive. That restricted state space, called the envelope, is used to expand an initial policy, extend the state space, and calculate a new policy [9]. There is no guarantee that an improved policy will appear without the need to include the entire state space.

- Monte Carlo - MDP solutions can be approximated by using a Monte Carlo approximation to model the next state based on the previous states. By keeping track of the transitions and rewards, estimations can be made to make educated guesses at the next states and their rewards.

Along with optimal solutions, there are many ways to approximate a solution to an MDP, which can often be much faster and get "close enough" to an optimal solution. Most popular solutions involve the discretization of continuous states, fixing the time horizon, prioritization, and approximating over the space with a granularity to sufficiently reduce the number of states.

2.11 Trust

Formally, trust is defined as "a charge or duty imposed in faith or confidence or as a condition of some relationship" and "something committed or entrusted to one to be used or cared for in the interest of another" [3]. This can easily be extended to an autonomous system of agents. Working in a multi-agent environment, it is easy

to assume that all of the other agents in the system are working to achieve the same goals as you and that none of them would subvert your attempts at success. As in real life, this is not always the case. Opponents in the system can be openly against your goals or operate more covertly in a way to deceive you of their real intentions. It is important to model the other agents in the system and decide if they are helping or hurting you with their actions.

There are many ways to determine trust in systems, including but not limited to

- Experience-based trust [5] - depends on the agent's past experiences with each other agent to determine its current trust rating.
- Reinforcement learning with trust [41] - works similarly to other reinforcement learning models. Agents receive reward based on how the other agents in the system perform. Trust and reward is gained if an agent expected another agent to do something, and it was indeed done. Trust and reward are lost when the agent performs differently than expected.
- Trust vectors [38] - allow for different levels of trust in which an agent may be trustworthy to do one thing, but untrustworthy to do another.
- The Trust-Based I-POMDP [44] - allows for agents to model both deceitful and unknown agent behaviors, and decide in a partially observable and stochastic environment what the best course of action is. Trust is gained through cooperation and proper modeling of the other agents and lost through betrayal and poor estimation of the other agent's actions.

2.12 Summary

This research is an amalgamation of many research areas, as such, an understanding of many different elements of artificial intelligence is required. Network

communication must be modeled realistically, as well agent trust. The environment the agents within it must be modeled by some form of MDP to allow an algorithm-based solution, and the multiple objectives must be taken into consideration. Once a solution is produced, it must be evaluated against the optimal solution or the current best solution, to determine its worthiness as a true solution.

Optimizing anything requires an understanding of not only the environment and problem to be optimized, but also the actual means of optimization. In turn, multi-objective optimization requires the understanding of multiple environments, problems, and optimization methods.

III. Area Defense With the Non-Dominated Sorting Genetic Algorithm II

To automate an area defense system, the systems task agents to perform a set of jobs. Each job is a smaller component of the larger goal of defending an area. These smaller components include area exploration for proper defense and communications maintenance for deconfliction of tasks and proper trust maintenance.

This chapter describes a task assignment GA to task agents to perform area exploration and communications maintenance as a constraint satisfaction problem. The algorithm uses the stochastic nature of NSGA-II to position agents in ways that meet the prescribed fitness functions. The solution is an organization of agents in 2-space that provides for the best exploration and communication at any one time. It focuses on a proof of concept to keep overhead down and achieve the required goals. The actual planning of the agents' actions, how an agent gets from point A to point B, is not considered, only how the agent's final position effects the system as a whole.

The term "agent" will be used to describe a component of this system, rather than as an actor with percepts and actuators as described in Chapter 2. Only the Behavior-based simulation uses true agents, all other simulations use the term agent to refer to the part of the simulation that is not the environment, a simulation characteristic, and does not remain static.

3.1 Framework Development

A true area defense system must operate in physical space, taking into effect all three physical dimensions. Agents in the system must make decisions about where to move, what to believe based on their percepts of the environment and other agents, and how to act based on those beliefs. Any of these decisions can be broken down into

smaller subsystems, which rely on specific protocols; exploration and movement are governed by actuator controls and state machines, communication links are facilitated by physical hardware and software protocols. Because this is a concept demonstration, most of these fine details are abstracted away to only the description of the effect in order to maintain a reasonable problem space and facilitate generalities based on two important aspects of these systems.

The following sections describe the limits of the MDP-based models and why they were not chosen to solve this problem. It also describes the CSP-based solution used to solve this problem, as well as the formulation of the CSP for solving with NSGA-II, including gene structure and fitness functions.

3.2 Limits of MDPs

One of the greatest limitations of any MDP-based model is the inability to describe multiple objectives in the reward functions. A model can include multiple objectives, for instance a reward for reaching a location on the world map which has not been explored recently, as well as a reward for linking up with a large number of agents, but this does not achieve a true multi-objective solver. For each state/action pair, the action with the single greatest reward will be chosen (often called the greedy case) and the existence of multiple objectives is never considered. An agent considering an action for time step t_i could consider an action in support of objective A as the best for that time step, and in the time step immediately following t_{i+1} consider an action in support of objective B as the best. Depending on the nature of the objectives, this could result in an agent policy which moves back and forth between two states ad infinitum.

Artificially biasing the reward values results in the same greedy policy. If action a_i results in a reward of 5 for objective A and a reward of 9 for objective B, and an

action a_j results in a reward of 10 for objective A and a reward of 4 for objective B, any weighting of the rewards, be it 50%-50%, 10%-90%, will always result in only one objective considered. In this situation, it becomes necessary solve the problem away from the normal MDP framework, utilizing a CSP to solve a multi-objective problem.

3.2.1 Domain and Environment.

A true area defense system needs to function in all three physical dimensions, but simulating this physical space is very computationally intensive. This section discusses the multi-agent simulation developed, and its key components.

Depending on the complexity of the simulation, from any one point in 3-space there are an infinite number of directions and distances in which an agent can move. Typically this is reduced to a more realistic approximation by assuming a lack of flight, or an inability for the agent to move in a direction perpendicular to the current plane. For this simulation, the domain is abstracted down to a two-dimensional grid world, in which agents occupy an X-Y Cartesian location on the grid. This allows the agents to use Chebyshev distance [54] to move in the grid world. Chebyshev distance allows for movement consistent with a two-dimensional grid as well as ease of distance calculation in that it forces distances to remain whole numbers, and be representative of the number of movements one location is from another. Rather than using the traditional distance formula, the number of grid cells separating two locations is used. This grid world can be of arbitrary size and shape if so desired. The grid world size is a parameter in testing.

Each agent has a specific observation and communication radius (r) associated with it. An agent in location (X_a, Y_a) can observe any location (X_b, Y_b) if and only if

$$\sqrt{(X_a - X_b)^2 + (Y_a - Y_b)^2} \leq radius.$$

This observation radius is the simulated equivalent of the range of a local sensor array, and allows the agent to know with a probability what is within its observation range.

The communication radius represents its physical counterpart. This is the range at which agents on the grid can successfully send messages to one another. This model for communication takes into account the necessary aspects of wireless network research [28], but abstracts away any networking protocols at high or low level that would bog down simulation. Messages can either be passed, or they cannot. Outside this communication radius, messages cannot be passed, but inside, there is certainty that information is passed with success. The observation radius and communication radius can be one in the same, if the designer so desires. Separating the observation radius and communication radius will alter the simulation, and likely produce a situation in which the agents move between radii, maximizing their communication at one moment, and maximizing their observed space at the next.

The domain in test contains no walls, but if they were present, they could be made to have an effect on the communication and observation radii at the developer's discretion. These effects would follow the tenets of wireless network research [28] and would be implemented by decreasing observation distance or communication distance across a wall. Again, these two modifiers could be separately defined between observation and communication.

The agents themselves only maintain two percepts. The first is their observed space, represented in the grid world by their observation radius. The second is a list of neighbors and connected agents. The list of neighbors is the identifying number of all agents that are within the communication radius of that specific agent, and the list of connected agents is the compiled list of neighbors' neighbors. If an agent A is connected to agent B, and agent B is connected to agent C, then regardless of agent A's neighbor status with agent C, it is connected. This is determined using a

depth-first search.

The tenets of wireless research, as discussed in the previous chapter, are applied (or abstracted away) in this research in the following ways:

- *The world may not be flat.* In this case, a two-dimensional environment, it is flat.
- *A radio's transmission area is not necessarily circular.* This research assumes transmission area is indeed circular, to allow
- *All radios do not have equal range.* All of these simulations assume homogeneous agent capabilities. This can easily be modified to suit another researcher's needs.
- *Symmetry in communication may not always exist.* This is similar to the previous point.
- *Perfect communication is not guaranteed.* Since this research focuses on establishing a link, not the error correction or quality associated with it, then it can be assumed that communication is perfect if it exists at all.
- *Signal strength is not always a function of distance.* For this set of simulations, signal strength is completely a function of distance. This keeps the state description simple and still capture useful information.

3.2.2 The Constraint Satisfaction Problem.

The problem being solved is the best distribution of the robots in the domain that optimizes the two constraints of area observation and communications maintenance across the agents.

Defining a grid world and state representation within the bounds of a CSP is a matter of defining the rules for the grid world and agent locations themselves. These

constraints are both a function of the environment, as well as the fitness functions associated with the state (population individual).

This type of problem domain defined as a CSP has very specific constraints associated with the grid-world itself:

- Agents cannot occupy or observe space outside the grid world. Assuming a square world: $\forall a : (X_a, Y_a) < size(world) AND (X_a, Y_a) > -1$ where X and Y are components of Cartesian coordinates.
- Two agents cannot occupy the same X-Y coordinate. $\forall a, b : X_a \neq X_b AND Y_a \neq Y_b$ where X and Y are components of Cartesian coordinates.

In the specific CSP definitions established in [41], $V(n)$ represents a set of n X-Y Cartesian coordinate pairs, one for each of the n agents; this is defined as:

$$\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}.$$

The constraint set C is a combination of the above constraints:

$$\forall a : X_a, Y_a < size(world) \cap X_a, Y_a > -1$$

and

$$\forall a, b : X_a \neq X_b \cap Y_a \neq Y_b$$

as well as the constraints defined in the Sections 3.2.5- 3.2.8 using genetic fitness functions.

The domain D for each of these variable pairs is a Cartesian coordinate in which:

$$\forall x, y : x, y \in Z^+.$$

The base constraints and domain above simply provide for proper indexing into the grid world. To be valid, grid assignments must be whole numbers, non-negative numbers, and no larger than the size of the grid. Also, two agents cannot share the same grid assignment.

3.2.3 Using NSGA-II.

In using a genetic or evolutionary algorithm to solve a CSP, the CSP must be presented to the algorithm in a way that is consistent with the algorithm's design. In the case of NSGA-II [10], and the software package used, JNSGA2 [33], an individual member of the population is described as the grid coordinate assignments for each of the agents. Fitness takes on several forms, as discussed below, but the JNSGA2 package works in minimization, meaning the a fitness value closer to 0 is better, with negative fitness values not allowed. Each population member has an observation value and a communication value as discussed above. These values are assigned uniformly at the simulation start, and remain static across generations.

In an attempt to diversify the way exploration is calculated, three slightly different exploration fitness functions were developed and tested against a common communication fitness function. These three exploration fitness functions are called "MaxDistance", "MaxArea", and "Tiled" respectively, and are described later in this chapter.

3.2.4 Gene Structure.

This particular problem establishes a solution as X and Y coordinate values for each agent in the system. An individual in the population is defined as the 2-space Cartesian coordinates of a set number of agents n,

$$\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}.$$

Gene mutation changes the value of either the X-coordinate, the Y-coordinate, or both. Gene crossover exchanges the coordinates of an agent from one gene with the coordinates of the same agent from another gene with some probability. Mutation of gene

$$\{(x_a1, y_a1), (x_a2, y_a2)\}$$

could result in

$$\{(x_a3, y_a3), (x_a4, y_a4)\}$$

where

$$P(x_a1 = x_a3) <= mutationProb$$

$$P(x_a2 = x_a4) <= mutationProb$$

$$P(y_a1 = y_a3) <= mutationProb$$

$$P(y_a2 = y_a4) <= mutationProb$$

Crossover of genes

$$\{(x_a1, y_a1), (x_a2, y_a2)\}$$

$$\{(x_b1, y_b1), (x_b2, y_b2)\}$$

would result in the new genes

$$\{(x_b1, y_b1), (x_a2, y_a2)\}$$

$$\{(x_a1, y_a1), (x_b2, y_b2)\}$$

where

$$P(VariableSwap) <= crossoverProb.$$

All gene mutation and crossover functions use their respective NSGA-II setting for probability and Java's Random class, which generates random numbers on a uniform distribution.

As long as the X and Y values meet the constraints above, that is, they fall within the domain of the grid world, then they are considered legal solutions to the CSP. Adding additional constraints is done in the form of fitness functions for the NSGA-II algorithm.

3.2.5 Distance Maximization.

For this type of area defense problem, it becomes a natural solution to place the agents as far away from each other as is possible to allow for the least possible overlap in searched space. The first exploration function simulates this by creating the largest average distance between all agents.

Because JSNGA2 works to minimize the solution, the average distance returned is subtracted from the greatest possible average distance the agents can be from each other, that is, with one agent being in a corner, and the other agent being in the opposing corner. If grid world of 1000x1000 is assumed, so the returned average value is subtracted from $1000 * \sqrt{2}$. This constraint is described in detail below, where P and Q represent agents in the environment.

$$AvgDist = \forall P, Q \text{ in } a \ \Sigma(distance(P, Q))/A^2$$
$$Fit_1 = 1000 * \sqrt{2} - AvgDist$$

3.2.6 Covered Area Maximization.

Another way to simulate area exploration is to keep track of the amount of area covered by all agents at any one time. In an area defense problem such as this, keeping track of the amount of area each agent can cover is similar to a human patrol

responsible for protecting a building or a watchtower responsible for protecting as far out in all directions as its observation will allow.

The second exploration function tries to minimize the number of unobserved locations on the grid world. The agent's radius is used to calculate the number of locations on the grid that can be observed by each agent, this is then totaled and returned as a percentage of the total space. A space that is completely observed returns a value of 0, and a space that is completely unobserved returns a value of 1.

$$ObservedArea = \Sigma(2\pi * r_A) - overlapingArea$$
$$Fit_2 = 1 - (ObservedArea/TotalArea)$$

The most fit solution is one in which no agents share any observed space, that is, each space that is observed, is observed by only one agent.

3.2.7 Tiled Area Coverage.

In a human area defense exercise, it is common for a unit to be assigned a specific section of the environment to control or maintain presence over. This section may be larger than the observation area of the unit. This could easily occur in an automated defense system, and is simulated by a tiled environment.

The third exploration function uses a tiled space to spread the agents around. The grid world is divided into uniform tiles, one tile for each agent. This is done by dividing the space into n evenly sized rectangles. The position of the agent then determines if a tile is explored or covered. In a 100x100 grid world with 4 agents, each agent is responsible for a 50x50 square of the grid. The agent may not necessarily be able to observe the entire grid section, but just as a human responsible for patrolling a space, the agent itself could then decide how best to patrol its assigned area.

Divide into $size(a)$ tiles

$$AvgDist = \forall P \text{ in } a \text{ determine which tile } P \text{ is in and mark covered}$$

$$Coverage = \Sigma(CoveredTiles)$$

$$Fit_3 = size(a) - Coverage$$

The number of uncovered tiles is returned as the fitness, with a fitness of 0 representing the ideal solution where all tiles have an agent covering them, and A representing a solution in which no tile has an agent in it. A fitness of A is impossible within the constraints of the problem, but does accurately capture an absolute worst case scenario. The worst case scenario within the bounds of the CSP would be $A-1$, in which all agents are grouped within a single tile.

The shapes and sizes of the tiles can be chosen by the researcher, but for this research, tiles are evenly sized rectangular pieces of the grid world. These tiles are generated by dividing the grid evenly in half vertically, and then dividing those halves horizontally to achieve the desired number of tiles.

3.2.8 Connectivity Fitness.

The fitness for the connectivity objective uses a set connectivity radius to determine which agents form a MANET. The radius r is an abstraction of the transmission distance of wireless hardware, with any agent inside the radius considered connected, and any agent outside the radius considered unconnected. Neighbors of neighbors are considered connected, forming the MANET. The fitness of an gene is returned as the number of independent networks, that is, the number of distinct groups of agents. For a problem with A number of agents, a solution in which all agents can communicate with all other agents returns the value 1, and a solution in which no agents have any neighbors returns the value a.

For a group of n agents, the least fit solution would return n, representative of no connections between any agents, or n distinct network clusters. The most fit

Algorithm 2 Connection Fitness

```
1: procedure Calculate Connection Fitness
2: Create DN_i, list of direct neighbors for each agent i
3: for each agent i do
4:    add i to DN_i
5:    for each agent j do
6:       if distance(i, j) ≤ r then
7:          ∪(DN_i, DN_j)
8:       end if
9:    end for
10: end for
11: for j < A^2 do
12:    for i < A do
13:       for each DN_k do
14:          if DN_k contains i then
15:             ∪(DN_k, DN_i)
16:          end if
17:       end for
18:    end for
19: end for
20: Remove duplicates from DN
21: Return size(DN)
22: end procedure
```

solution is always 1, which represents a single network cluster, in which every agent is connected to the network, and can communicate with every other agent via a direct link, or a series of connected neighbors.

3.3 Agent Behaviors

As a comparison against a simple baseline algorithm, a reflex agent is developed in which agents move in 2-space in random directions looking for a fixed gateway, an immobile agent which acts as the root for a network tree. The agent action is to select a random direction and move in it until a world boundary is reached or a link is observed. If a world boundary is reached, a new random direction is chosen. If a link is observed, the agent's reward goes to 1 and the agent stops moving. The

agent then observes the neighbors it can connect with and if a cycle is detected, using a depth-first search, its reward goes to 0 and the agent begins to move again. The ultimate goal is to reduce the redundancy of the overall network, and spread the agents across the largest space possible while still maintaining a network and a link with the gateway. The agents should spread themselves out as far as they can and still maintain a connection to the gateway. Convergence is defined as every agent in the system achieving a reward of 1. When convergence is reached, the average distance between the agents is recorded.

3.3.1 The Dudek Taxonomy.

The above domain and environment description can be compared directly with the taxonomy laid out by Dudek, et al. [11]. For this problem, SIZE-LIM is appropriate, as the number of agents is definitely more than 0, and likely more than 2. SIZE-LIM also allows the number of agents to scale relative to the size of the environment. Next, COM-NEAR is used to describe the communications capabilities of the agents, in that they can only communicate with agents "that are sufficiently nearby". TOP-ADD can be used to describe the communications protocol used by the agents which allows them to communicate directly with another agent without the need for a hierarchy or broadcasting to all agents within range. This particular element can be modified at the user's request, and does not have an effect on this simulation. BAND-INF describes the free cost of communication in this simulation. Because networking protocols and overhead are ignored completely, communication is effectively free. Communication in more accurate simulation, or the applied physical system would have a cost associated with it. ARR-DYN describes the dynamic nature of the network topology. As this is a MANET-type system, the physical layout of the network nodes (the agents) can be completely different from one time step to the next. PROC-TME

describes the agents' computation models as Turing Machine Equivalents. This is again not important for this simulation, but would be used in the physical construct. COMP-IDENT describes the individual agents in the system as being identical. This is an abstraction specifically for this simulation, as a true physical design could be made up of many different robots with differing computational and problem-solving capabilities; the agent that acts as a mobile network repeater will likely have different capabilities than an agent that acts as an attack and interdiction drone.

3.4 Summary

When utilizing a genetic algorithm to describe a constraint satisfaction problem, it becomes necessary to define the CSP in a way that can be properly utilized by the GA. In this case, the variable assignments V become the gene for each member of the GA's population, and the associated domain for V, D, as well as the constraints C become integrated into the gene structure and the fitness functions associated with the specific problem. In this situation, constraints such as the legality of agent positions on the grid are handled in the gene structure, and constraints like communication maintenance are handled in the fitness functions.

IV. Simulation Testing and Results

The evolutionary algorithm-based constraint satisfaction problem was tested using one communication fitness function and three different exploration fitness functions. The NSGA-II settings, as well as environment specific parameters were tested using one of the three fitness functions. The following describes both the parameter testing, as well as the tests of the three exploration fitness functions against the single communication maintenance fitness.

All three fitness functions showed consistency in the need to sacrifice area observability for communication maintenance, with "MaxArea" and "Tiled" being the most effective. In the case where observation was better than communication, maximizing communication meant limiting observation, and in the case where communication was better than observation, the improved communication helped, but ultimately still required a sacrifice in observation to maximize communication.

Because of the randomness introduced by the mutation and crossover schemes, the search contour for all of these simulations is very rough. Two adjacent solutions or population members could be vastly different due to the highly stochastic nature of generating position assignments for each population member.

4.1 Testing

All fitness function testing was performed using the same environment parameters. These parameters were chosen arbitrarily and verified using tests against modified parameters. The testing used the parameters described in Table 2. The parameter-specific testing utilized the "MaxArea" fitness function and the default parameter settings (Table 1) unless otherwise noted.

Generational testing was planned, however increasing the number of generations

Table 1. Default Simulation Parameters.

Parameter	Value
Generations	100
λ	100
Number of Agents	10
Mutation Probability	90
Crossover Probability	90
Grid Size	1000x1000
Communication Radius	100
Explore Radius	100

an order of magnitude increased the simulation time by two orders of magnitude and it was decided that this change was unnecessary. Simulations showed that after 100 generations, sufficient Pareto optimal solutions were being generated such that solutions were being repeated often. Adding an order of magnitude to the number of generations would have resulted in 4-day simulations with no performance improvement. It was also planned to test a larger grid space, but system memory restrictions prevented it. A grid world of 10000X10000 would require just over 38GB RAM, 2000X2000 would require 1.5GB, more than the Java Development Environment (JDE) has available to it. A smaller grid space was considered, but any substantial reduction in grid size (an order of magnitude) resulted in complete or nearly complete coverage by 10 agents, so this result was ignored. These results are described in the following section, but ultimately had no effect on the trend of the gathered data.

4.2 Parameter Test Results

The following subsections describe the altered parameters for the simulation testing, as well as the trends in their results. Each of the results gathered from parameter changes is consistent with a diametric opposition between area exploration and communications maintenance.

The default parameters for the tests are shown in Table 1 and utilized the "MaxArea" fitness function. Parameters were altered individually and compared to the final results for validity.

Table 2. Simulation Parameters.

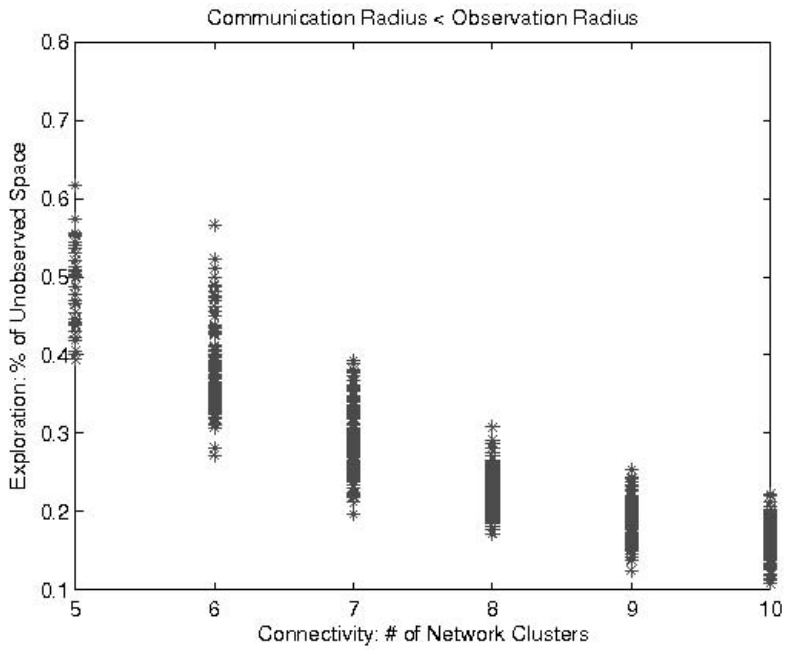
Parameter	Value
Generations	100
λ	100
Number of Agents	10, 20
Mutation Probability	45, 90
Crossover Probability	45, 90
Grid Size	1000x1000
Communication Radius	100, 200
Explore Radius	100, 200

4.3 Observation and Communication Tests

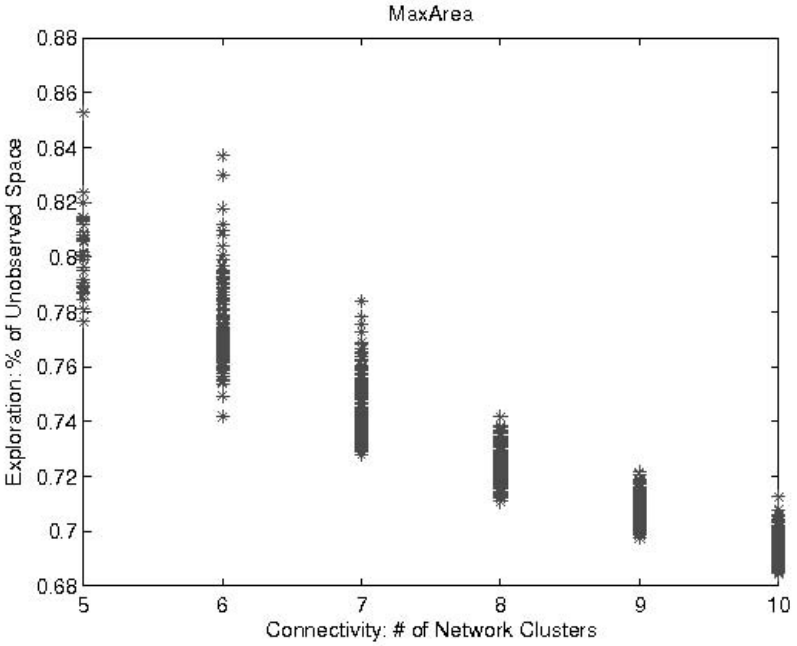
The tests shown in Figures 5 and 6 show why assuming observability and communication radii to be equal is acceptable in this problem. Improved communication delays the need to sacrifice exploration to maintain communication. Better communication is always desirable, but there will always be a point at which communication and exploration are no longer compatible without sacrifices to one of them.

Figure 5(a) utilizes a 200 grid unit observation radius and shows a system in which there is a need to sacrifice exploration to maintain communication. When compared to Figure 5(b) which uses a 100 unit observation, a similar trend is shown, but less of the area can be observed at once, which is consistent with a reduced observational capability.

Figure 6(a) shows a system in which the exploration radius (100 units) is less than the communication radius (200 units). When compared to Figure 6(b), similar trends are shown and data is still in agreement with the other results. In this simulation, agents have the ability to communicate farther than they can observe, which means



(a) 200 unit Observation Radius and 100 unit Communication Radius

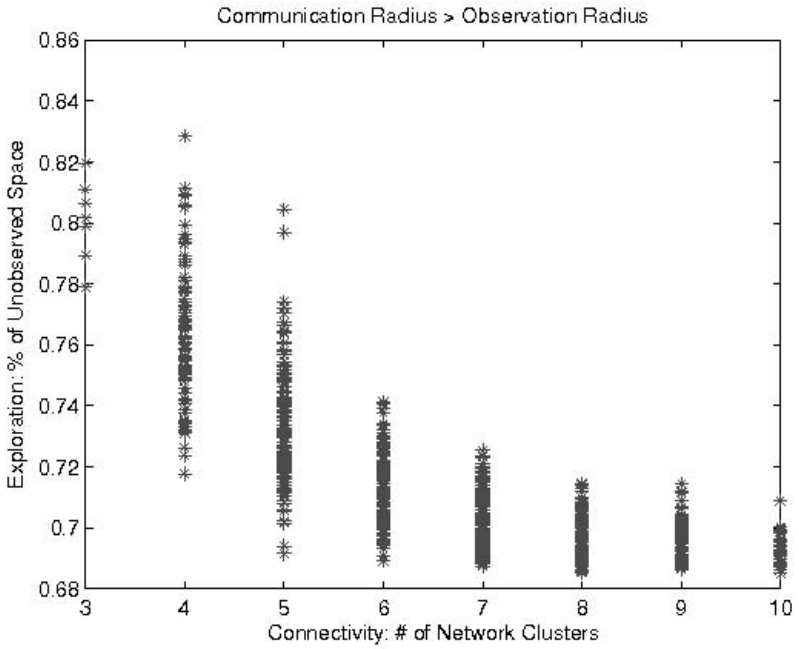


(b) 100 unit Observation and Communication Radius

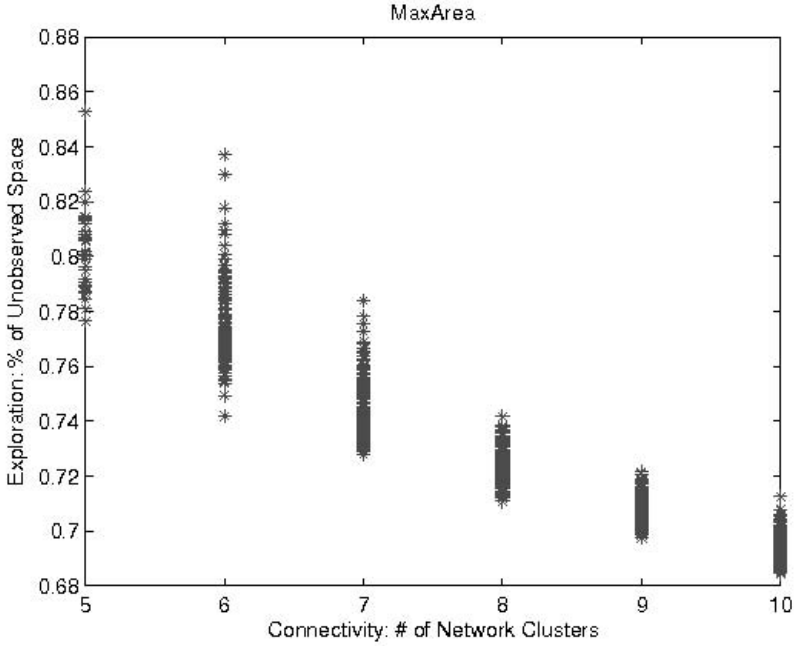
Figure 5. 200 unit Observation Radius and 100 unit Communication Radius

they must move farther before they need to disrupt communication to maximize area observation. The almost horizontal trend in the front from 5 network clusters to 10

shows that the increased communication radius helps, but only to a point. Maximizing communication requires a sacrifice in observed space.



(a) 100 unit Observation Radius and 200 unit Communication Radius



(b) 100 unit Observation and Communication Radius

Figure 6. 100 unit Observation Radius and 200 unit Communication Radius

Figure 7(a) shows a system in which both observational and communication capabilities are doubled. When compared to Figure 7(b), which uses a 100 unit observation radius, it shows a consistent trade off between exploration and communication. To achieve the most connected set of agents, with 3 distinct networks, it only becomes possible to observe about 55% of the environment. The next most connected set of agents, with 4 networks, allows for the observation of roughly 68% of the environment. The least connected networks, only one or 2 agents connected, allow for the most area observation, providing roughly 88% observation of the environment.

4.4 NSGA-II Parameter Test - Mutation and Crossover Probabilities

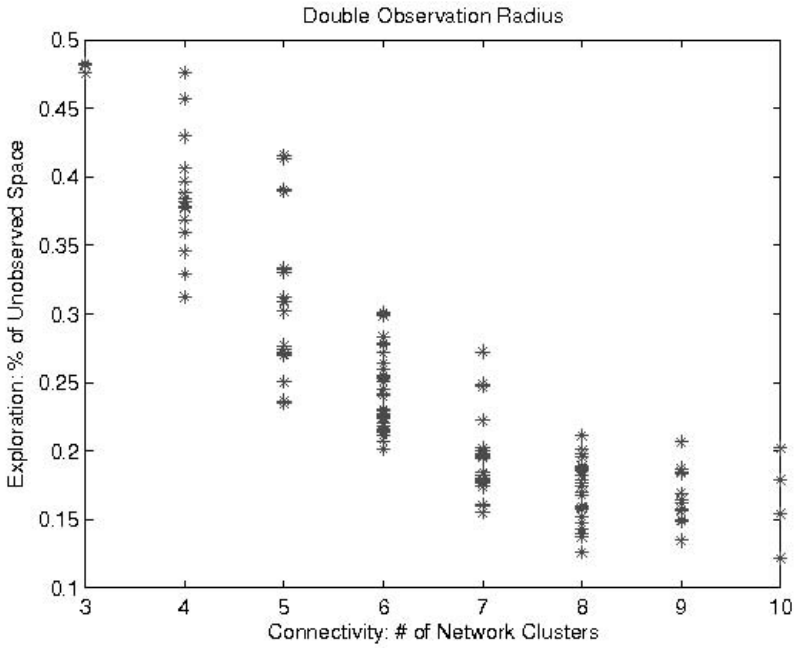
Figure 8(a) shows a simulation with 45% mutation and crossover probabilities. When compared to Figure 8(b), which uses 90% for its mutation and crossover probabilities, it shows the same trend, in which area exploration is sacrificed for communication maintenance, and communication maintenance is sacrificed for area exploration.

4.5 Fitness Function Tests

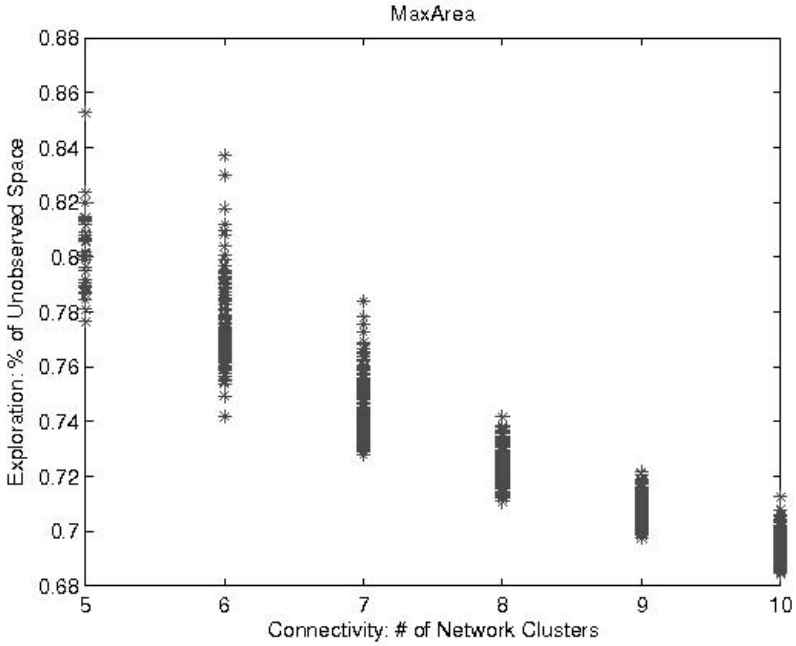
All fitness testing was done with a randomly generated population size of 100 individuals, over 100 generations, with mutation and crossover probabilities of 90%.

Each of the three exploration fitness functions, “MaxArea”, “MaxDistance”, and “Tiled” were tested against the same communication fitness function 30 times, and the results were aggregated together into one Pareto front for each fitness function. No visualization was developed for this simulation.

For each of the tests performed using NSGA-II, 30 separate iterations were run, with each iteration consisting of the 1000x1000 grid world, 10 agents with a uniform 100 unit radius, 100 generations each with a population size of 100, and both



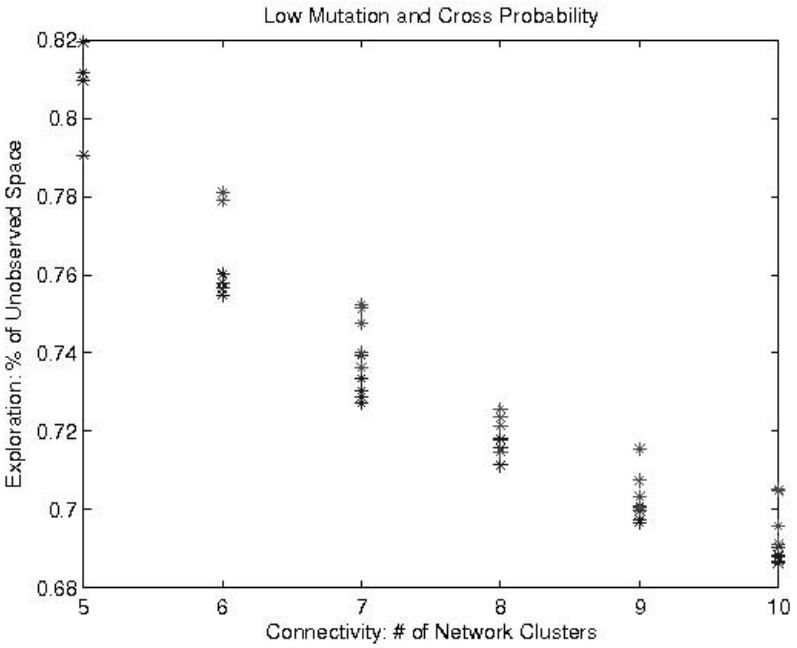
(a) 200 unit Observation and Communication Radii



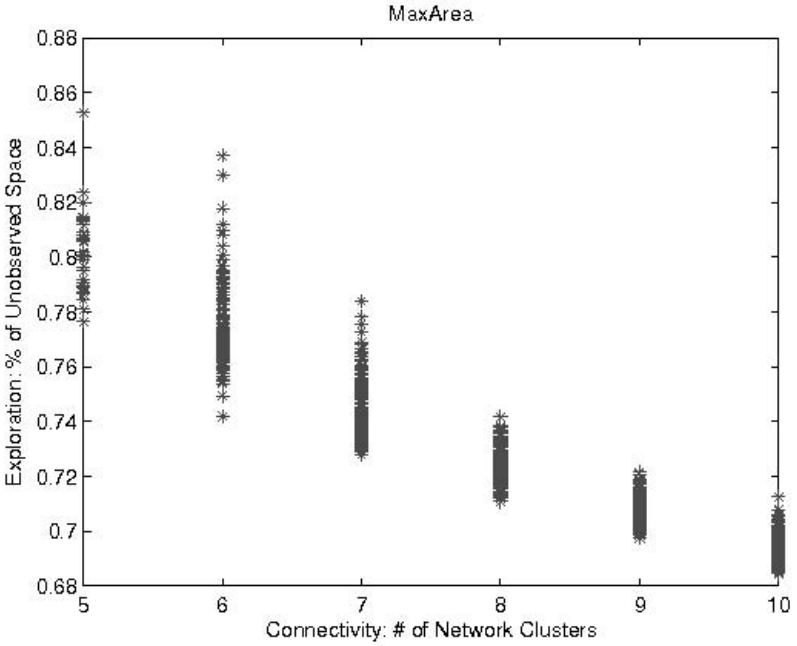
(b) 100 unit Observation and Communication Radius

Figure 7. 200 unit Observation and Communication Radii

a crossover and mutation probability of 90%. The population size and number of generations were chosen with respect to the available computing power, being the



(a) 45% Mutation and Crossover Probabilities



(b) 90% Mutation and Crossover Probabilities

Figure 8. 45% Mutation and Crossover Probabilities

limits that the computers available could solve in a reasonable amount of time (several hours). As stated earlier, NSGA-II has a computational complexity of $O(M * \lambda^2)$

for each generation, where M is the number of objectives (2), and λ is the population size (100).

The crossover and mutation probabilities of 90% are completely arbitrary, with the hopes of introducing the most diversity into the population. These parameters were tested against 45% mutation and crossover, as shown in section with no effect on the final solution trends; convergence to a Pareto front was slower, but the trend of the front itself remained the same. At each generation, the non-dominant solutions were recorded and used as the seeds for the next generation. All of the results for each of the 30 iterations were combined into one solution. As stated earlier, connectivity for each is measured as the number of independent networks, with fewer being better and representative of a larger connectivity among all agents.

The following describes the results of the individual fitness function tests. It shows consistent opposition between the two objectives of area exploration and communications maintenance no matter the metric used to determine area exploration.

4.5.1 Maximizing Distance.

This experiment (Figure 9) measured exploration as the average distance between all of the agents in the grid world, and returned the average distance as the best case minus the average, resulting in a solution closer to 0 being the better solution.

These results clearly show maximizing the distance between the agents has a negative impact on the communication capabilities, and vice versa. A linear relationship is expected, but because the return value is average distance between all agents and not the individual distances between a set of agents, the resulting front has some concavity associated with it.

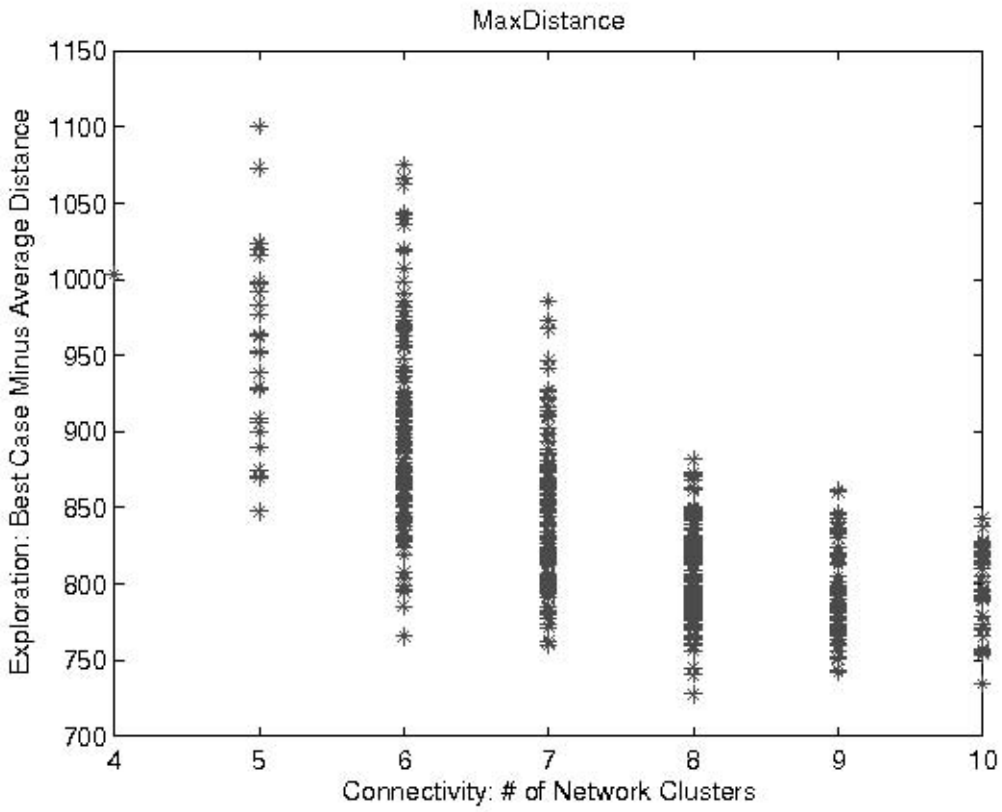


Figure 9. The Pareto Front in which exploration is measured using average distance from each agent to each other agent.

4.5.2 Maximizing Area.

This experiment (Figure 10) measured exploration as the percentage of unobserved locations on the grid world, with a lower percentage representative of less overlap in the area observed by the agents and more total space being observed.

The Figure 10 results show an almost linear relationship between maximizing the observed area and maximizing the connectivity. It can be asserted that there is a true linear relationship based on the error introduced into the calculation of the exploration. Observed area is calculated using a grid which is marked observed or unobserved using the communication radius and the location of the agent. It is filled using a drawing-style circle fill algorithm which approximates the circle, resulting in slightly different values depending on how rounding is performed.

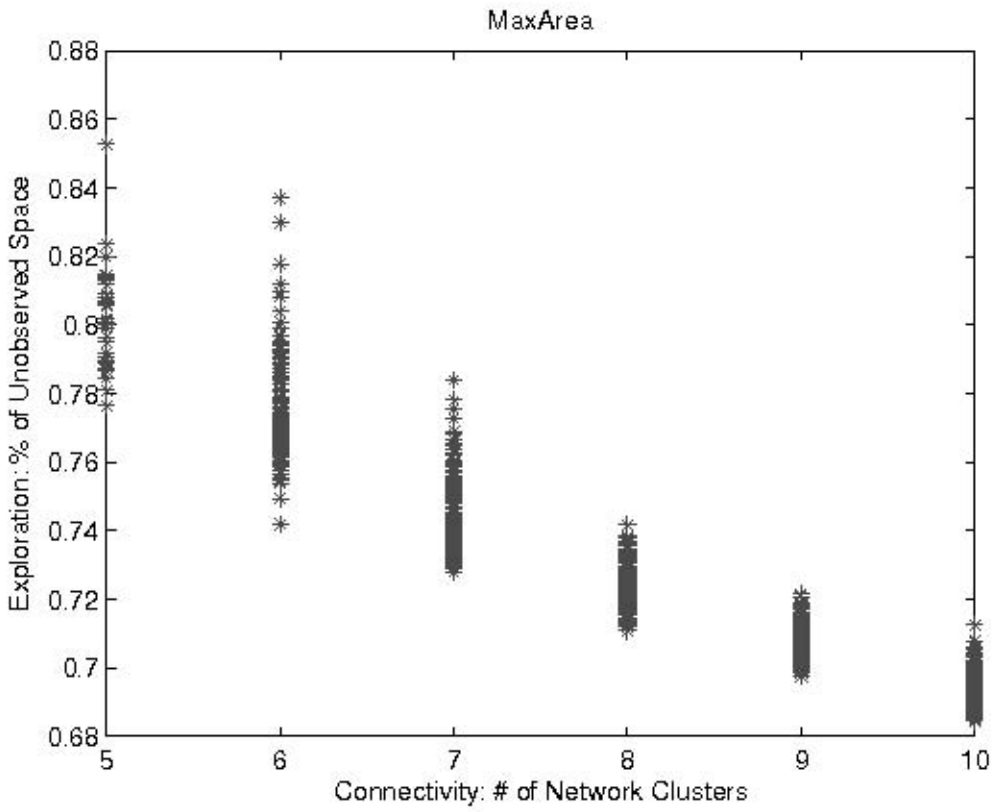


Figure 10. The Pareto Front in which exploration is measured using the amount of unobserved space on the grid world.

4.5.3 Tiled Space.

This experiment (Figure 11) measured exploration by dividing the grid world into even sections, one for each agent. If a tile has an agent in it, it is considered covered, and fitness is measured by the number of uncovered tiles.

The result of the tiled space clearly shows the direct conflict between spreading out to cover all of the tiles, and maintaining a large network. In order to cover the most tiles, with 10 agents on a 1000×1000 grid world, the agents must be positioned outside their communications radius. A tile is approximately 500x200, and with a communication radius of 100, the smallest number of network groups is 3; being 2 groups of 4 agents, and 1 group of 2 agents, positioned approximately in the corners, as shown in 12.

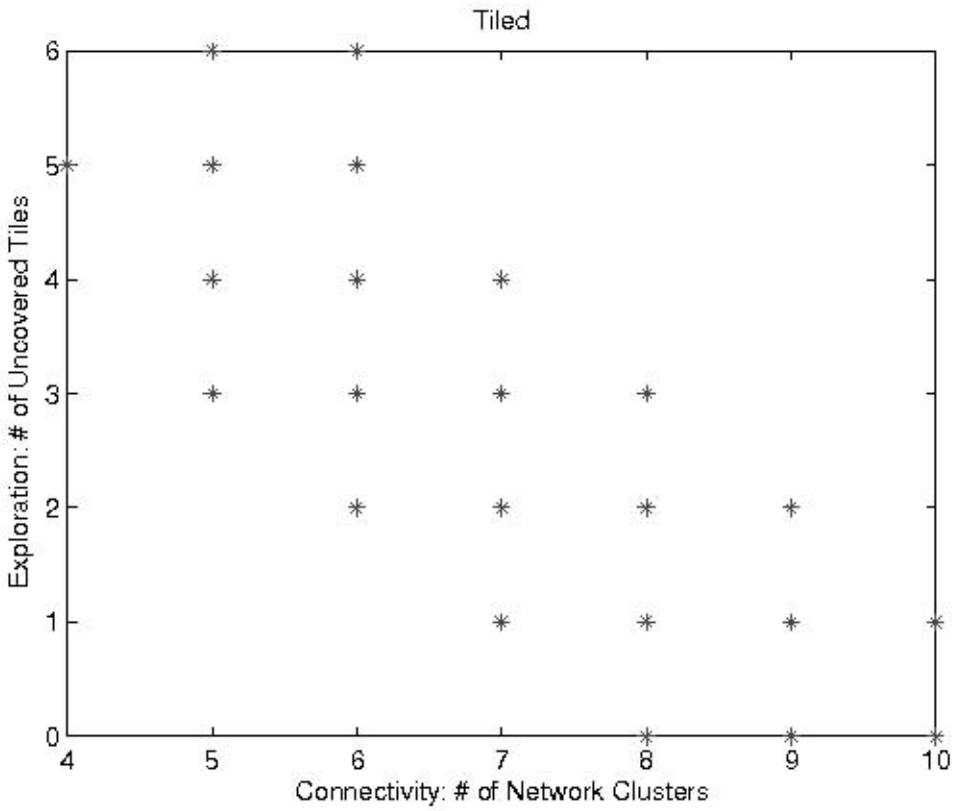


Figure 11. The Pareto Front in which exploration is measured using tiles which are either covered or uncovered.

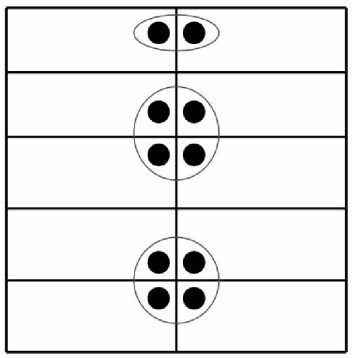


Figure 12. This shows one of the best solutions for the tiled exploration test; each of the tiles is covered with an agent and there are three distinct networks as shown in red.

4.6 Domain Altering Test - Number of Agents

Figure 13(a) shows the simulation results for a 1000x1000 grid world, with 20 agents using a 100 unit observation and communication radius and 90% mutation

and crossover probabilities. When compared to Figure 13(b), which uses only 10 agents, it shows a consistent trend in which area observation is directly sacrificed in order to maintain communication, and vice versa. The number of agents in the environment has an effect on the total observed space, but the same compromises are needed to maintain exploration and communication.

4.7 Behavioral Simulation Results

This behavioral based simulation was run with 100 separate iterations, in the same 1000x1000 grid world, with varying communication capabilities. The results in Tables 3 - 5 show a decrease in the average distance between an agent and its neighbors.

Table 3. Average Distances Between Agent and Neighbors, Comm Distance = 50

CommDist	50		
Without Spread	64.3083	63.2301	61.734
With Spread	61.3381	59.8581	57.8583

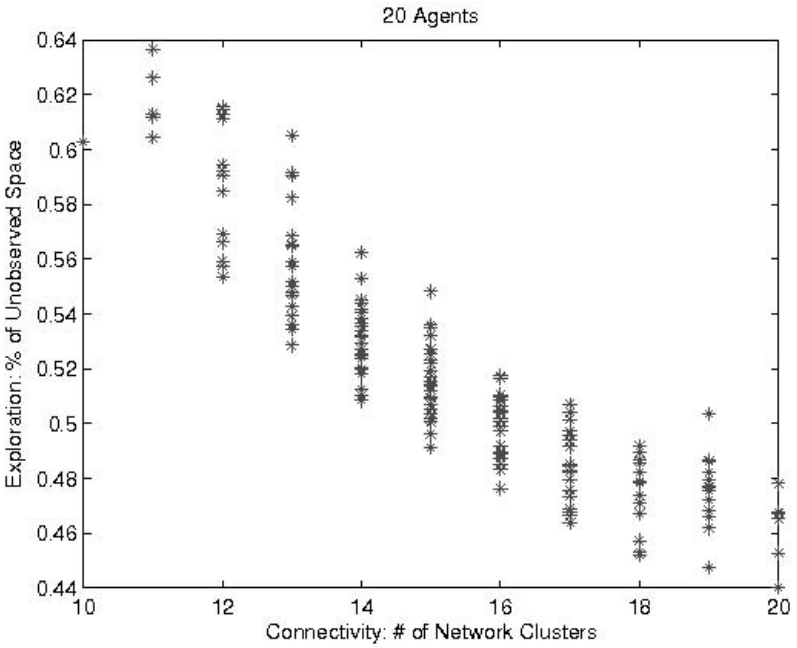
Table 4. Average Distances Between Agent and Neighbors, Comm Distance = 150

CommDist	150		
Without Spread	183.4816	183.9411	199.752
With Spread	170.0364	168.6085	168.4061

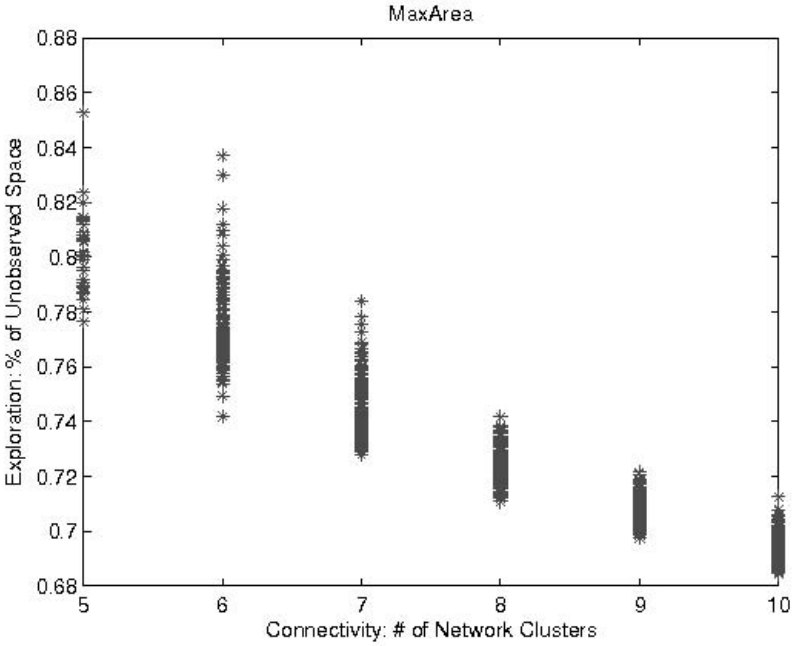
Table 5. Average Distances Between Agent and Neighbors, Comm Distance = 300

CommDist	300		
Without Spread	335.4667	362.69	568.848
With Spread	286.4364	290.0939	296.2495

Initially, it appears that the spreading capability did not work, but this is not the case. The spreading function is the cycle detection in the neighbor list, forcing each agent to move to a location where it only has 1 or 2 connections, representative of an end node (one neighbor) or center node (two neighbors). This puts the agent



(a) 20 agents in the environment



(b) 10 agents in the environment

Figure 13. 20 Agents in the Environment

at the absolute limits of its communication, slightly closer to its neighbors for better communication, but farther away from more agents. This provides for a more uniform

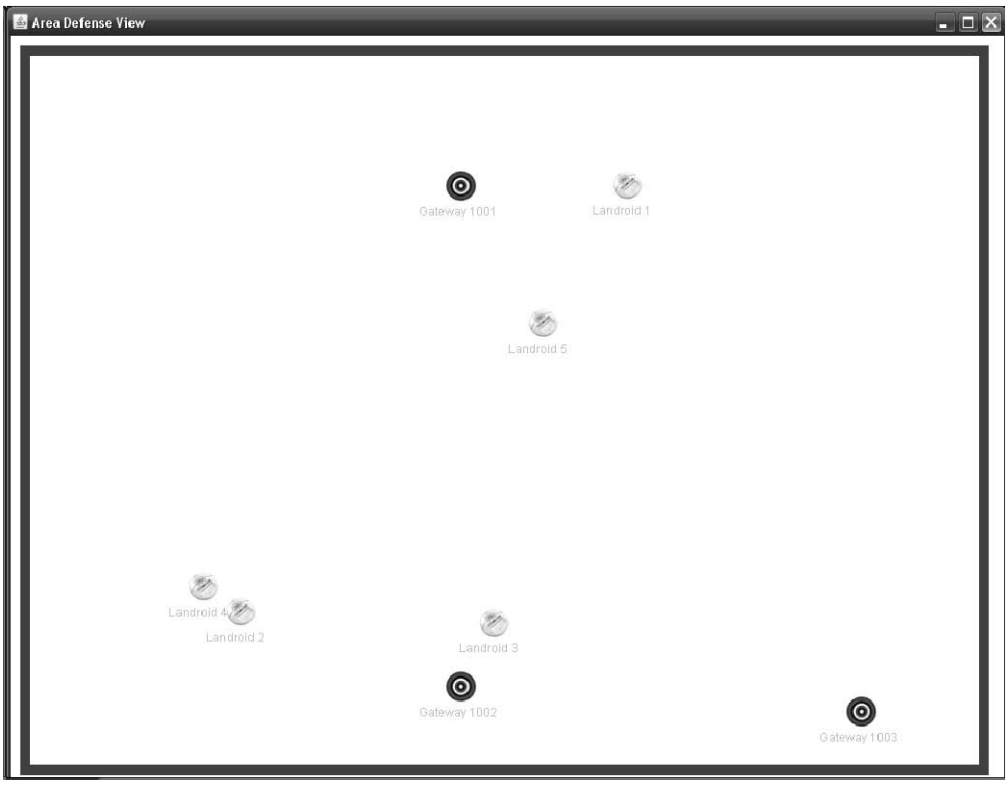


Figure 14. An example of the visualization of the behavior-based simulation.

distribution of agents within the environment. Agents are not as spread out, but they are more ideally placed in that they do not overlap as much space, and do not position themselves in a location that makes their contribution to the entire network redundant and useless.

This is consistent with the GA test results, in that the ideal solution for an agent looking to maximize both its observed area or maximum average distance from all other agents is to position itself at the very edge of its communication radius.

V. Conclusions

This research formulated a constraint satisfaction-based approach to solving a multi-agent positioning system with multiple competing objectives, utilizing the genetic algorithm NSGA-II to solve the CSP. As part of continuing trust research, it is necessary for groups of agents to maintain communication with each other in order to maintain trust.

5.1 Summary

Area exploration and group communications maintenance were described and solved in a fully distributed and uncoordinated way and show that exploration and communications maintenance, without any element of coordination, are diametrically opposed goals. Moving away from a group to explore requires the sacrifice of losing communication, while moving to the group to communicate limits the exploration of the space. Results consistently show high area exploration made up of keeping average distances between agents, covering the most unobserved space, or observing the most tiled zones always resulted in limited communication, with stronger communication, fewer individual networks, always resulting in less observation. In the cases where observation and communication radii were different, the simulations clearly showed this limitation. When observation is better than communication, approximately 60% area observation results in 5 different network clusters, and approximately 90% area observation results in no network at all, with all 10 agents unconnected from each other. When observation is weaker than communication, approximately 23% area observation results in 3 distinct networks and good connectivity, and approximately 38% area observation results in no network at all. In between those extremes lie several compromises that are consistent with agents moving back and forth trying to

optimize one goal over the other.

It then becomes necessary to implement some form of coordination, either in planning exploration routes, creating rules about maintaining certain distances from other agents (such as keeping agents at the edge of their observation or communication range), or moving in a group.

5.2 Future Work

The intended direction of this work was to integrate a multi-objective evolutionary algorithm with a traditional DEC-POMDP solver, to allow modeling of the multi-objective nature of the area defense problem. This solver would have then been tested against a more traditional DEC-POMDP solver in which the two objectives were artificially biased against each-other in an attempt to achieve a similar result.

Due to oversights made by the researcher in looking at this problem, only a static case was considered, in which no evolving agent models were developed, and the balance between agent exploration and agent communication was determined in a non-dynamic way.

Anyone looking to pick up this research should begin by modifying the environment to make it more complex. This can be achieved by adding walls, heterogeneous agent capabilities, and other elements that would limit observation and communication in the environment. Research should also look at development of a proper DEC-POMDP model for this area exploration and communication maintenance problem and focus on making the decision making process for the DEC-POMDP easier. This can probably be done with swarm or flocking coordination techniques, or artificially prioritizing the goals and rewards based on the environment itself. Using centralized exploration planning may also become necessary, as complete distribution is a challenge yet to be undertaken.

This research, specifically the information contained in the Pareto fronts, can be used by a researcher to decide how they wan exploration and communication to relate to each other for their simulation. They could decide that keeping the agents at a specific distance from each other is ideal, or they could decide that one simulation could focus on exploration, while another focuses on communication.

Bibliography

[1] "The Network Simulator ns-2", 2001. URL `http://www.isi.edu/nsnam/ns/`.

[2] "JiST/SWANS - Java in Simulation Time / Scalable Wireless Ad hoc Network Simulator", 2004. URL `http://jist.ece.cornell.edu/`.

[3] "trust. Merriam-Webster Online Dictionary". Feb 2010. URL `http://www.merriam-webster.com/dictionary/trust`.

[4] Bäck, Thomas. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, Oxford, NY, 1996.

[5] Barber, K. Suzanne and Joonoo Kim. "Belief Revision Process Based on Trust: Agents Evaluating Reputation of Information Sources". *Trust in Cyber-societies, LNAI 2246*, 73–82. Springer, 2001.

[6] Boutilier, Craig. "Planning, Learning and Coordination in Multiagent Decision Processes". *In Theoretical Aspects of Rationality and Knowledge*, 195–210. Morgan Kaufmann, 1996.

[7] Cao, Y. Uny, Alex S. Fukunaga, and A. B. Kahng. "Cooperative Mobile Robotics: Antecedents and Directions". *Autonomous Robots*, 4:226–234, 1997.

[8] Coello Coello, Carlos A., Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, NY, 2nd edition, 2007.

[9] Dean, Thomas, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson. "Planning Under Time Constraints in Stochastic Domains". *Artificial Intelligence*, volume 76, 35–74. Elsevier Science, Providence, RI, 1995.

[10] Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. "A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II". *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2000.

[11] Dudek, Gregory, Michael R. M. Jenkin, and David Wilkes. "A taxonomy for multi-agent robotics". *Autonomous Robots*, 3:375–397, 1996.

[12] Durillo, Juan J., Antonio J. Nebro, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. *jMetal: A Java Framework for Developing Multi-Objective Optimization Metaheuristics*. Technical Report ITI-2006-10, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, E.T.S.I. Informática, Campus de Teatinos, December 2006.

[13] Eiben, A. E., J.I. van Hemert, E. Marchiori, and A. G. Steenbeek. "Solving Binary Constraint Satisfaction Problems using Evolutionary Algorithms with an Adaptive Fitness Function". *Proceedings of the 5th Conference on Parallel Problem Solving from Nature, number 1498 in LNCS*, 196–205. Springer, 1998.

[14] Eiben, A.E. and J.E. Smith. "What is an Evolutionary Algorithm?" *Introduction to Evolutionary Computing*, 15–36. Springer, Cambridge, MA, 2nd edition, 2007.

[15] Ferentinos, Konstantinos P. and Theodore A. Tsiligiridis. "Adaptive design optimization of wireless sensor networks using genetic algorithms". *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 51(4):1031–1051, 2007.

[16] Fonseca, Carlos M. and Peter J. Fleming. "An Overview of Evolutionary Algorithms in Multiobjective Optimization". *Evolutionary Computation*, 3:1–16, 1995.

[17] Gerkey, Brian P. and Maja J Mataric. "Sold!: Auction methods for multi-robot coordination". *IEEE Transactions on Robotics and Automation, Special Issue on Multi-robot Systems*, 758–768, 2001.

[18] Gmytrasiewicz, Piotr J. and Prashant Doshi. "A Framework for Sequential Planning in Multi-Agent Settings". *Journal of Artificial Intelligence Research*, 24:24–49, 2004.

[19] Goldman, Robert P., David J. Musliner, Mark S. Boddy, Edmund H. Durfee, and Jianhui Wu. "Unrolling Complex Task Models into MDPs". Proceedings of the American Association for Artificial Intelligence, 2007.

[20] Guo, Yi, Lynne E. Parker, and Raj Madhavan. "Collaborative Robots for Infrastructure Security Applications". *Proceedings of the International Symposium on Collaborative Technologies and Systems*, 185–200. 2007.

[21] Hauskrecht, Milos and Branislav Kveton. "Approximate Linear Programming for Solving Hybrid Factored MDPs", 2006.

[22] Ist, Ines Lynce, Ins Lynce, and Jol Ouaknine. "Sudoku as a SAT Problem". *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics, AIMATH 2006*. Springer, 2006.

[23] Jang, Myeong-Wuk, Amr Ahmed, and Gul Agha. "Efficient Agent Communication in Multi-Agent Systems", 2004.

[24] Jin, Yaochu, Tatsuya Okabe, and Bernhard Sendhoff. "Adapting Weighted Aggregation for Multiobjective Evolution Strategies". *Proceedings of First International Conference on Evolutionary Multi-Criterion Optimization, Leture Notes in Computer Science*, 96–110. Springer, 2001.

[25] Jourdan, Damien B. and Olivier L. de Weck.

[26] Jr., Ed Pegg and Eric W. Weisstein. "Sudoku". MathWorld - A Wolfram Web Resource. URL `http://mathworld.wolfram.com/Sudoku.html`.

[27] Kang, Chih-Wei and Jian-Hung Chen. "Multi-objective evolutionary optimization of 3D differentiated sensor network deployment". *GECCO '09: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference*, 2059–2064. 2009.

[28] Kotz, David, Calvin Newport, and Chip Elliott. *The Mistaken Axioms of Wireless-Network Research*. Contract, Dartmouth College of Computer Science, July 2003.

[29] Marin, Olivier, Pierre Sens, Jean pierre Briot, Zahia Guessoum, and Bp Le Havre Cedex. "Towards Adaptive Fault Tolerance For Distributed Multi-Agent Systems", 2009.

[30] Mataric, Maja J., Gaurav S. Sukhatme, and Esben Astergaard. "Multi-robot Task Allocation in Uncertain Environments". *Autonomous Robots*, 14(2):255–263, 2003.

[31] Mausam, Daniel, and Daniel S. Weld. "Solving Relational MDPs with First-Order Machine Learning". *In Proceedings ICAPS Workshop on Planning under Uncertainty and Incomplete Information*. 2003.

[32] Mehta, Neville. "Divide-and-Conquer Methods for Solving MDPs". Oregon State University, 2006.

[33] Melcher, Joachim. *JNSGA2: Java Non-Dominated Genetic Sorting Algorithm II*. Technical report, Intitut AIFB, Universiaet Karlsruhe (TH), 2007.

[34] Moore, Brandon J. and Kevin M. Passino. "Distributed Coordination Strategies for Wide-Area Patrol". *Journal of Intelligent and Robotic Systems*, 56(1–2):23–45, 2009.

[35] Mostofi, Yasamin. "Decentralized Communication-Aware Motion Planning in Mobile Networks: An Information-Gain Approach". *Journal of Intelligent and Robotic Systems*, 56(1-2):233–256, 2009.

[36] Parsopoulos, K. E. and M. N. Vrahatis. "Particle Swarm Optimization Method in Multiobjective Problems". *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002)*, 603–607. ACM Press, 2002.

[37] Pastore, Tracy Heath, H. R. Everett, and Kevin Bonner. "Mobile Robots for Outdoor Security Applications". *Proceedings of the American Nuclear Society 8th International Topical Meeting on Robotics and Remote Systems*. 1999.

[38] Ray, Indrajit and Sudip Chakraborty. "A Vector Model of Trust for Developing Trustworthy Systems". *European Symposium on Research in Computer Security*, volume 3193, 260–275. Springer, 2004.

[39] Rendl, Andrea. "N-Queens Problem". TAILOR: Tailoring Constraint Models to Solvers, August 2009. URL `http://www.cs.st-andrews.ac.uk/~andrea/tailor/nqueens.png`.

[40] Rojas, Mara Cristina Riff and Rioe Rojas. "From quasi-solutions to solution: An Evolutionary Algorithm to solve CSP". *In Proceedings of the Principles and Practice of Constraint Programming Conference (CP96*, 367–381. Springer-Verlag, 1996.

[41] Russell, Stuart J. and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2003.

[42] Schillo, Michael, Bettina Fley, Michael Florian, Frank Hillebrandt, and Daniela Hinck. "Self-Organization in Multiagent Systems: From Agent Interaction to Agent Organization", 2002.

[43] Seuken, Sven and Shlomo Zilberstein. "Formal models and algorithms for decentralized decision making under uncertainty." *Autonomous Agents and Multi-Agent Systems*, 17(2):190–250, 2008.

[44] Seymour, Richard. *The Trust-Based Interactive Partially Observable Markov Decision Process*. Master's thesis, The Air Force Institute of Technology, 2009.

[45] Seymour, Richard and Gilbert L. Peterson. "A Trust-Based Multiagent System". *Proc. of the 2009 IEEE Int. Conf. on Information Privacy, Security, Risk and Trust (PASSAT '09)*, 109–116. IEEE Computer Society, 2009.

[46] Simmons, Reid, David Apfelbaum, Wolfram Burgard, and Dieter Fox Mark Moors. "Coordination for multi-robot exploration and mapping". *Proceedings of the AAAI National Conference on Artificial Intelligence*, 852–858. AAAI, 2000.

[47] Sosic, Rok. "A Parallel Search Algorithm for the N-Queens Problem". *IEEE Transactions on Systems, Man, and Cybernetics*, 21, 1994.

[48] Sutton, Richard S. and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.

[49] Wang, Chenggang, Saket Joshi, and Roni Khardon. "First Order Decision Diagrams for Relational MDPs". *Journal of Artificial Intelligence Research*, volume 31, 431–472. Tufts University, Medford, MA, 2008.

[50] Weiss, Gerhard. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 2000.

[51] Williams, Martyn. "Robots Take Dangerous Jobs", April 2003. URL `http://www.pcworld.com/article/110127/robots_take_dangerous_jobs.html`.

[52] Wooldridge, Michael. *An Introduction to Multiagent Systems*. John Wiley and Sons, Liverpool, UK, 2002.

[53] Zeng, Xiang, Rajive Bagrodia, and Mario Gerla. "GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks". *Proceedings of the 12th Workshop on Parallel and Distributed Simulations – PADS '98*, 154–161. 1998.

[54] Ziviani, Artur, Serge Fdida, Jos F. De Rezende, Otto Carlos, and M. B. Duarte. "Toward a measurement-based geographic location service". *Proceedings of PAM2004, Antibes Juan-les-Pins*, 43–52. 2004.

[55] Zou, Yi and Krishnendu Chakrabarty. "Sensor deployment and target localization in distributed sensor networks". *ACM Transactions on Embedded Computing Systems (TECS)*, 3(1):61–91, 2004.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
12–03–2010	Master's Thesis	Sept 2008 — Mar 2010

4. TITLE AND SUBTITLE

Multi-Objective Constraint Satisfaction
for Mobile Robot Area Defense

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Mayo, Kenneth W., 2d Lt, USAF

5d. PROJECT NUMBER

ENG09-219

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GCE/ENG/10-03

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Labs, Sensors Directorate, Reference Branch
Attn: Dr. Jacob Campbell (jacob.campbell@wpafb.af.mil)
2241 Avionics Circle, Area B, Bldg 620
Wright-Patterson AFB, OH
(937) 785-6127x4154

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In developing multi-robot cooperative systems, there are often competing objectives that need to be met. For example in automating area defense systems, multiple robots must work together to explore the entire area, and maintain consistent communications to alert the other agents and ensure trust in the system. This research presents an algorithm that tasks robots to meet the two specific goals of exploration and communication maintenance in an uncoordinated environment reducing the need for a user to pre-balance the objectives. This multi-objective problem is defined as a constraint satisfaction problem solved using the Non-dominated Sorting Genetic Algorithm II (NSGA-II). Applying the algorithm to the area defense problem, results show exploration and communication without coordination are two diametrically opposed goals, in which one may be favored, but only at the expense of the other. This work also presents suggestions for anyone looking to take further steps in developing a physically grounded solution to this area defense problem.

15. SUBJECT TERMS

constraint satisfaction, multi-objective, evolutionary algorithm

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Gilbert L. Peterson
U	U	U	U	83	19b. TELEPHONE NUMBER *(include area code)* (937) 255–3636x4281, gilbert.peterson@afit.edu